Spring 5-18-2018

Optimization of GPU-Accelerated Iterative CT Reconstruction Algorithm for Clinical Use

Tao Ge
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS
School of Engineering and Applied Science
Department of Electrical and Systems Engineering

Thesis Examination Committee:
Joseph A. O'Sullivan, Chair
David G. Politte
Ulugbek Kamilov

Optimization of GPU-Accelerated Iterative CT Reconstruction Algorithm for Clinical Use

by

Tao Ge

A thesis presented to the School of Engineering
of Washington University in St. Louis in partial fulfillment of the
requirements for the degree of

Master of Science

May 2018

Saint Louis, Missouri

Acknowledgments

I would like to thank my father, Ming Ge, my mother, Dan Ye and my grandmother, Huiyun Zhu, for everything they gave me, including their love, patience, suggestion and support.

I thank all of the educators I have had the pleasure of learning from, for the knowledge they granted me.

I thank those graduate students who have been helpful during my studies.

I thank my colleagues in Dr. Joseph O'Sullivan's lab. They provided many useful ideas for my research.

I thank my advisors Dr. Joseph A. O'Sullivan and Dr. David G. Politte for their helpful instruction and continuous support.

Special thanks goes to the member of my committee for their time and comments and critique of this work: Dr. Joseph A. O'Sullivan, Dr. David G. Politte, and Dr. Ulugbek Kamilov.

Tao Ge

Washington University in St. Louis
May 2018

Contents

List of Figures

List of Tables

ABSTRACT OF THE THESIS

Optimization of GPU-Accelerated Iterative CT Reconstruction Algorithm for Clinical Use

by

Tao Ge

Master of Science in Electrical Engineering

Washington University in St. Louis, 2018

Research Advisor: Professor Joseph A. O'Sullivan

In order to transition the GPU-accelerated CT reconstruction algorithm to a more clinical environment, a graphical user interface is implemented. Some optimization methods on the implementation are presented. We describe the alternating minimization (AM) algorithm as the updating algorithm, and the branchless distance-driven method for the system forward operator. We introduce a version of the Feldkamp-Davis-Kress algorithm to generate the initial image for our alternating minimization algorithm and compare it to a choice of a constant initial image. For the sake of better rate of convergence, we introduce the ordered-subsets method, find the optimal number of ordered subsets, and discuss the possibility of using a hybrid ordered-subsets method. Based on the run-time analysis, we implement a GPU-accelerated combination and accumulation process using a Hillis-Steele scan and shared memory. We then analyze some code-related problems, which indicate that our implementation of the AM algorithm may reach the limit of single precision after approximately 3,500 iterations. The Hotelling observer, as an estimation of the human observer, is introduced to assess the image quality of reconstructed images. The estimation of human observer performance may enable us to optimize the algorithm parameters with respect to clinical use.

Chapter 1

Introduction

1.1 Motivation

X-ray computed Tomography (CT) plays an important role in current clinical diagnosis of many diseases and in treatment planning in radiation oncology, including proton therapy. X-ray CT integrates the measured data received by detectors from different views to produce a stack of images. Physicians use the image volume for diagnostic purposes and treatment planning purpose. The quality of the image volume impacts the reliability of the decisions that physicians make.

Our ultimate aim is to assess the performance of decisions that physicians make based on the quality of image volumes. In radiation oncology, these decisions impact the treatment that patients receive. In radiation oncology, one possible ideal task is that physicians treat patients using a treatment using different CT images produced by different algorithms, and then observe patients for years, possibly decades. However, it is not realistic to wait for so many years to get a result. A simplified task should be introduced for this assessment. This raises our first motivation: to start the study to assess whether images produced by our algorithm could lead to better diagnosis or treatment plans than algorithms currently used in the clinic.

Another goal is to help the project of dual energy CT (DECT) imaging for proton therapy. Proton therapy is a high-potential particle therapy treatment used to irradiate diseased tissue. Due to the high dose-distribution sensitivity of proton therapy used to tissue composition and electron density, researchers propose to implement an accelerated and optimized quantitative DECT mapping process and conduct a prospective virtual clinical trial to assess the performance of DECT used in proton therapy. Our work could provide a fundamental tool for acceleration and optimization of the DECT project.

A graphical user interface (GUI) is a useful component for a clinical researcher who is not familiar with the reconstruction algorithm and the integrated development environment. Initially, we want to implement a prototype of the GUI for our single energetic CT reconstruction code. This GUI will enable clinical researchers to utilize our fast AM algorithm for tree-dimensional (3D) CT without requiring detailed prior knowledge of the algorithms. Our implementation should be fast enough for clinical use and should set algorithm parameters according to the intent of researchers. Figure 1.1 is a graphical user interface for our single-energetic CT reconstruction which may be used by researchers in the future.

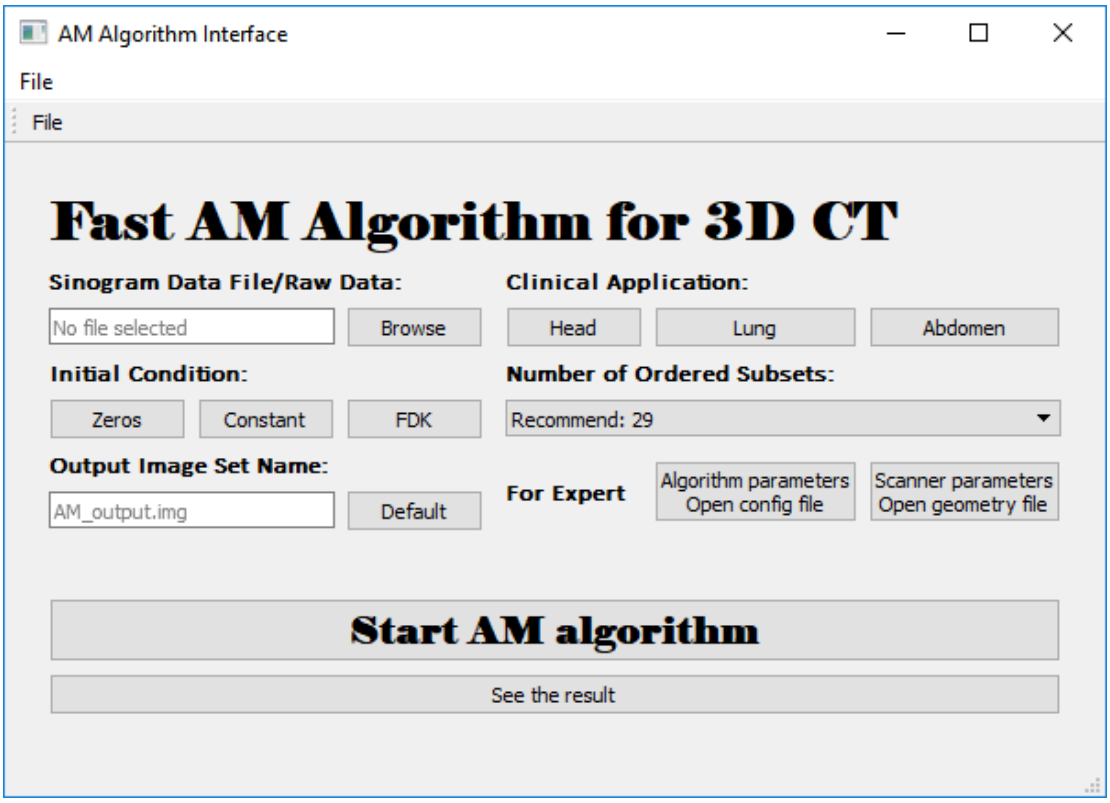


Figure 1.1: Graphical user interface for single-energetic AM CT reconstruction

Through this interface, researchers select the transmission data, choose the initial condition, set the output image name and the number of ordered subsets of the AM algorithm. The application will provide presets of several sets of optimized parameters, including number of iterations and scalars for the penalty term, with respect to clinical use. The result would be displayed in ImageJ [1], an image processing program designed for scientific multidimensional images.

1.2 Outline

This thesis has 6 chapters. Chapter 1 introduces the motivation for writing this thesis. In Chapter 2, we present the basic ideas of the updating algorithm, system operator, notation, geometry, data and the operating environment in our implementation. We introduce several acceleration methods in Chapter 3. In Chapter 4, we analyze several code optimization issues. In Chapter 5, a measure of image quality is introduced in ordered to optimize algorithm parameters and future planned experiment is discussed. Chapter 6 concludes this thesis with a discussion about future work.

Chapter 2

Background

In our CT reconstruction procedure, we use a branchless distance-driven method to get the system matrix and use an alternating minimization algorithm to update the image. This chapter introduces the branchless distance-driven method, the alternating minimization algorithm, the system geometry, parameters and GPU acceleration. The whole process of CT reconstruction is shown in Figure 2.1.

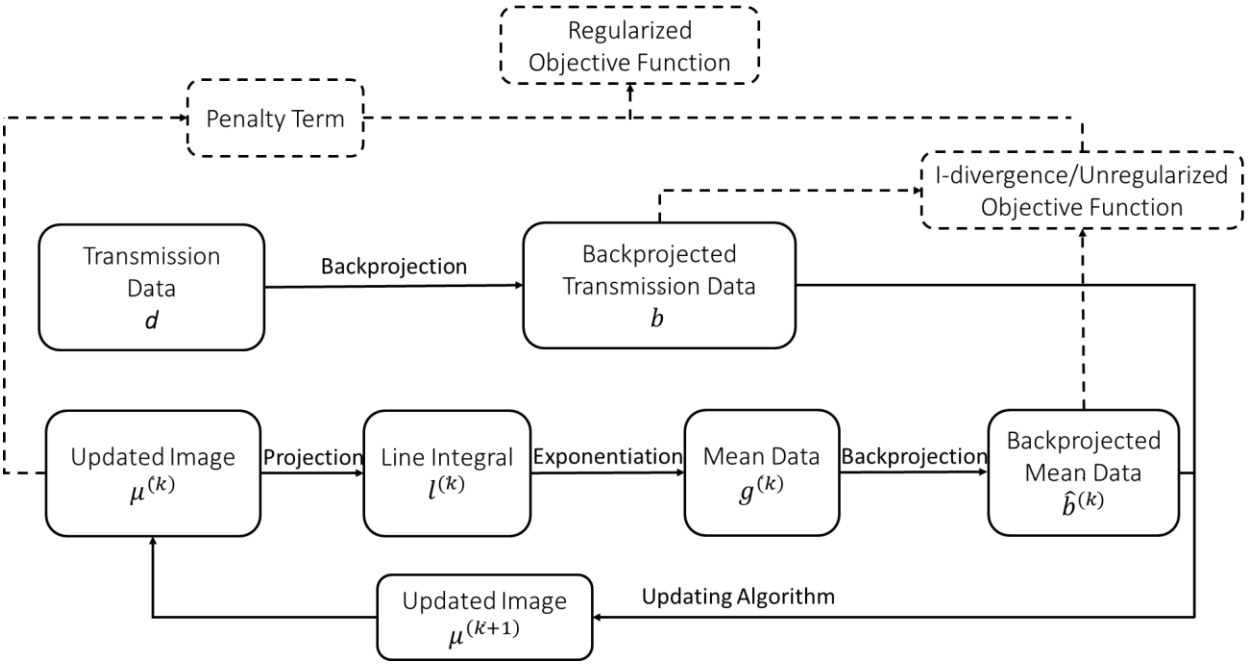


Figure 2.1: Flowchart of the CT reconstruction process

2.1 Methods and Algorithms

2.1.1 Branchless Distance-driven Method

The branchless distance driven method, derived by Samit Basu and Bruno De Man [2], is a highly parallelizable method for projection and backprojection. Compared to the original distance-driven

method (DD), the branchless DD divides the overlap computation into three operations: integration of the initial image, interpolation/anterpolation, and differentiation, which are completely decoupled. Therefore, we are able to accelerate the projection and back-projection procedures using parallel computation on GPUs.

The basic idea of the distance-driven method is allocating the value according to the relative position between detectors and voxels. Let V denote the image value, D denote the detector value, v denote the projected image edge and d denote the detector edge. Figure 2.2 is a sample of the distance driven method. The computation of detector values would be:

$$D_{1,2} = \frac{V_{1,2}(v_2 - d_1) + V_{2,3}(d_2 - v_2)}{d_2 - d_1}$$

$$D_{2,3} = \frac{V_{2,3}(v_3 - d_2) + V_{3,4}(d_3 - v_3)}{d_3 - d_2}$$

$$D_{3,4} = \frac{V_{3,4}(v_4 - d_3) + V_{4,5}(d_4 - v_4)}{d_4 - d_3}$$

$$D_{4,5} = \frac{V_{4,5}(d_5 - d_4)}{d_5 - d_4}.$$

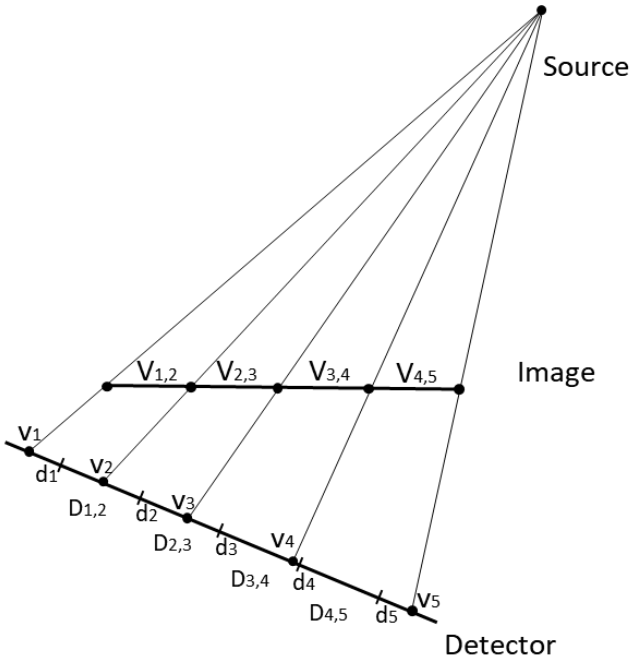


Figure 2.2: A sample of the distance-driven method

Since in the distance-driven method, irregular boundaries of projected voxels and detectors lead to poor predictability in the overlap kernel, the branchless distance-driven method was introduced as an advanced decoupled DD method for projection and backprojection. The process of the DD method can be written as a definite integral of the value of V from d_i to d_{i+1}, which is also equivalent to the difference between two definite integrals of V with intervals $(-\infty, d_{i+1}]$ and $(-\infty, d_i]$. According to [1], the DD process can be converted to an interpolation of accumulated data.

Then, the branchless DD method for projection/backprojection has three steps:

1) integrate/differentiate
2) interpolate/anterpolate
3) differentiate/integrate

2.1.2 Alternating Minimization (AM) Algorithm

The alternating minimization algorithm, derived by O'Sullivan and Benac [3], is a statistical iterative algorithm for estimating the attenuation-coefficients.

In order to get the optimal values of the reconstructed image, we want to find the image that minimizes objective function between our measured data and the mean data. Denoting the transmission data by d and mean data by g, the I-divergence for the mono-energetic case is

$$I(d||g) \triangleq \sum_i \left[d_i log\left(\frac{d_i}{g_i}\right) + g_i - d_i \right]$$

Where

$$g_i(\mu) = I_i e^{-\sum_j a_{ij}\mu_j}$$

Index j stands for the image-space coordinates, and index i stands for a pair of detector and source position. I_i is the mean number of source counts in the absence of an attenuating medium, μ_{ij} is the

attenuation coefficient of voxel j, and a_{ij} is the point-spread function map the detector domain and the image domain.

Then the backprojections required in the alternating minimization algorithm can be written as

$$b_j = \sum_i a_{ij} d_i$$

$$\hat{b}_j^{(k)} = \sum_i a_{ij} g_i\left(\mu^{(k)}\right)$$

where (k) indicates the iteration number. The update for attenuation coefficients of the alternating minimization algorithm defined by O'Sullivan and Benac (2006) is:

$$\mu_j^{(k+1)} \triangleq \max\left(\left[\mu_j^{(k)} - \frac{1}{Z}\log\left(\frac{b_j}{\hat{b}_j^{(k)}}\right)\right], 0\right)$$

The penalty term is defined as [4]

$$R(\mu) = \sum_j \sum_{j_n \in N(j)} w\,(j, j_n)\,\psi\left(\mu_j - \mu_{j_n}\right)$$

$$\psi(t) = \delta^2\left(\left|\frac{t}{\delta}\right| - \log\left(1 + \left|\frac{t}{\delta}\right|\right)\right)$$

where $N(j)$ is the set of neighboring voxels of voxel j, $w(j, j_n)$ is the weight calculated as the distance from voxel j to voxel j_n, and δ is an adjustable parameter of AM algorithm which controls the transition between the quadratic (for small $|t|$) and linear region (for large $|t|$).

The objective function is then a combination of I-divergence and the penalty term,

$$\Phi(\mu) = I(d||g(\mu)) + \lambda R(\mu)$$

where λ is a scalar controlling the strength of the penalty term.

The regularized attenuation coefficient μ_j is computed by solving [5]:

$$b_j - \hat{b}_j^{(k)} e^{z_j(\hat{\mu}_j - \mu_j)} + \lambda \sum_{j_n \in N(j)} w(j, j_n) \frac{\partial \psi(t)}{\partial t}\bigg|_{t=2\mu_j - \hat{\mu}_j - \hat{\mu}_{j_n}} = 0$$

Due to the complexity of the penalty term, Newton's method is used to solve this equation.

2.2 Geometry and Parameters

In this thesis, the modality of interest is multislice helical x-ray CT. Multislice helical x-ray CT has proven to be a successful imaging modality in many clinical applications. The source and the detectors rotate helically together around the body, and the source continuously projects x-ray photons through the object to the detectors. The figure below, plotted by Daniel Keesing [5], shows the basic structure of multislice helical x-ray CT.

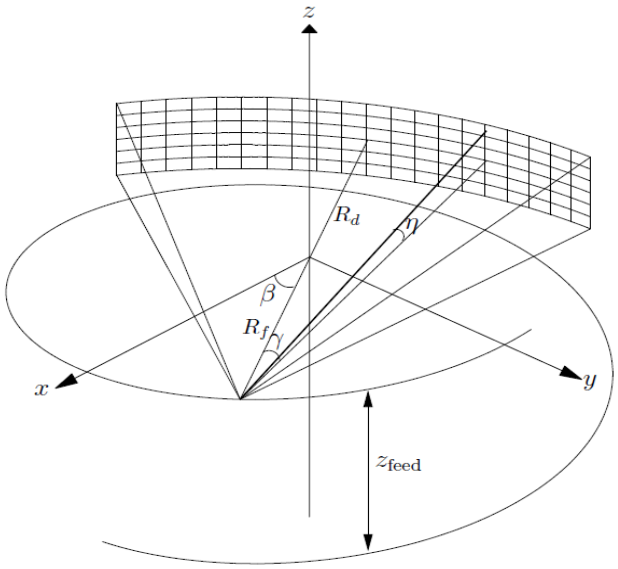


Figure 2.3: Simplified geometry of helical x-ray CT

Each individual detector is specified by the cone angle η and the fan angle γ. R_f is the distance between the focus of the x-ray source and the isocenter. the source rotation from the isocenter. R_d is the distance between the isocenter and the center detector. β is the angle between the positive x axis and the line connecting the focal spot to the isocenter. z_{feed} is the distance the bed moves during one rotation.

2.3 Data

Two sets of transmission data are utilized in this thesis. The first data set is a clinical chest-to-abdomen-scan of a pediatric patient acquired with a Siemens Sensation 16 scanner at St. Louis Children's Hospital. Table 2.1 shows the parameters of data acquisition, and Figure 2.4 and Figure 2.5 show the transmission sinogram and the reconstructed images for several slices of the clinical data, respectively. The second data set is simulated NCAT-phantom transmission data. Figure 2.6 shows several slices of the NCAT phantom to be projected.

Table 2.1: Parameters of Data acquisition

Siemens Sensation 16 scanner	
Transmission Data	Chest Scan on Child/ NCAT-Projected Data
Image volume size	512*512*164
Number of views	13920
Number of channels	672
Number of rows	16
Number of views per rotation	1160
Channel spacing	0.00135413
Row spacing	1.5
Source to center distance	570.0
Feed per rotation	24
x/y spacing	1.0 mm
z spacing	3.0 mm
Number of slices per rotation	12

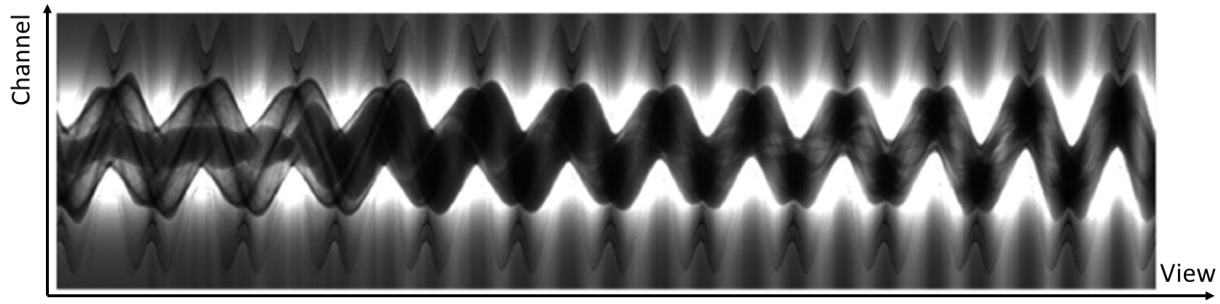


Figure 2.4: Transmission sinogram of the clinical data

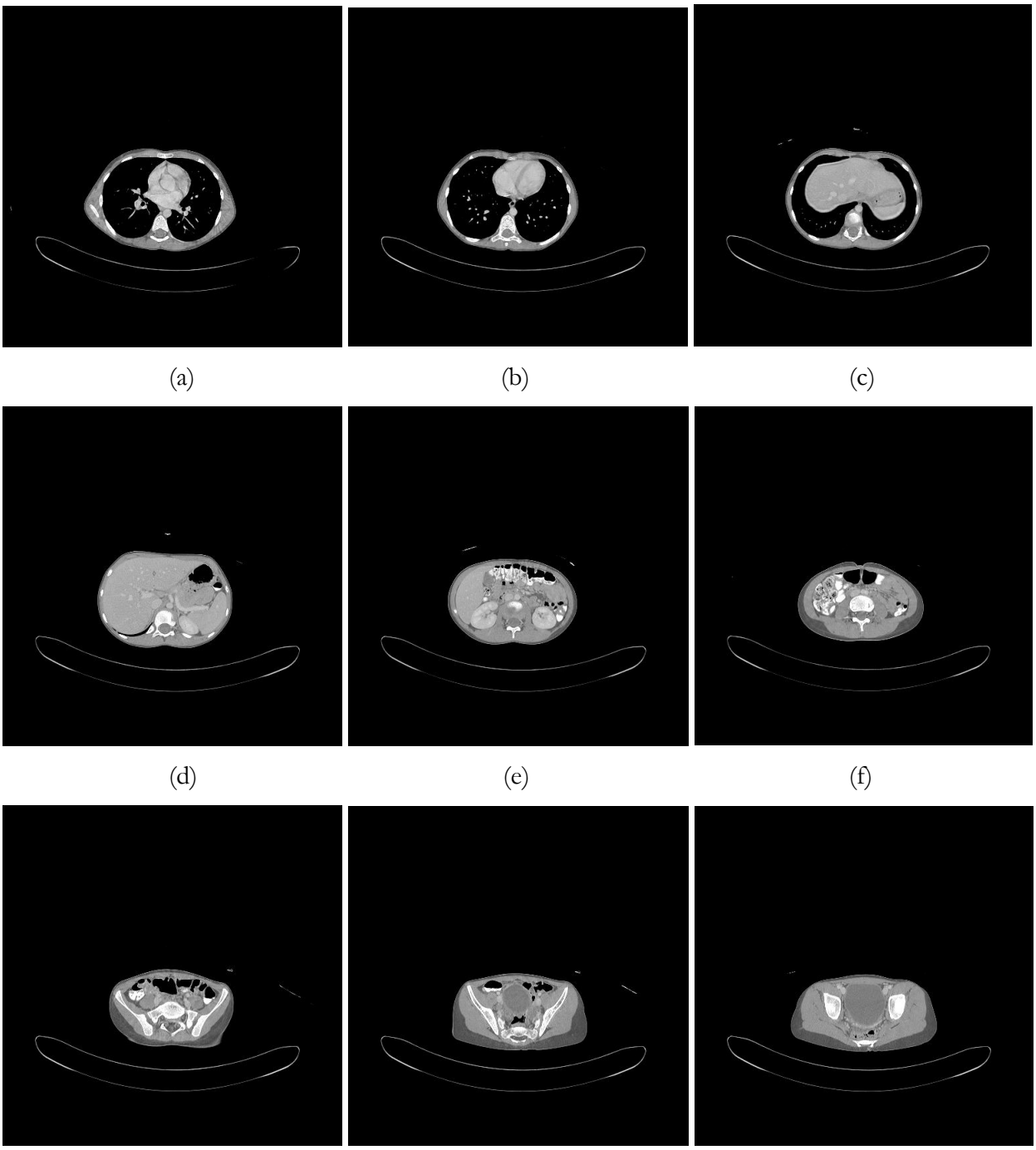


Figure 2.5: The reconstructed image of clinical transmission data for (a) The 12th slice (b) The 26th slice (c) The 41th slice (d) The 57th slice (e) The 81th slice (f) The 102th slice (g) The 123th slice (h) The 134th slice (i) The 147th slice. No order subsets, 4000 iterations, initial condition: zeros. Display window: [0.0129,0.0259] mm^{-1}.

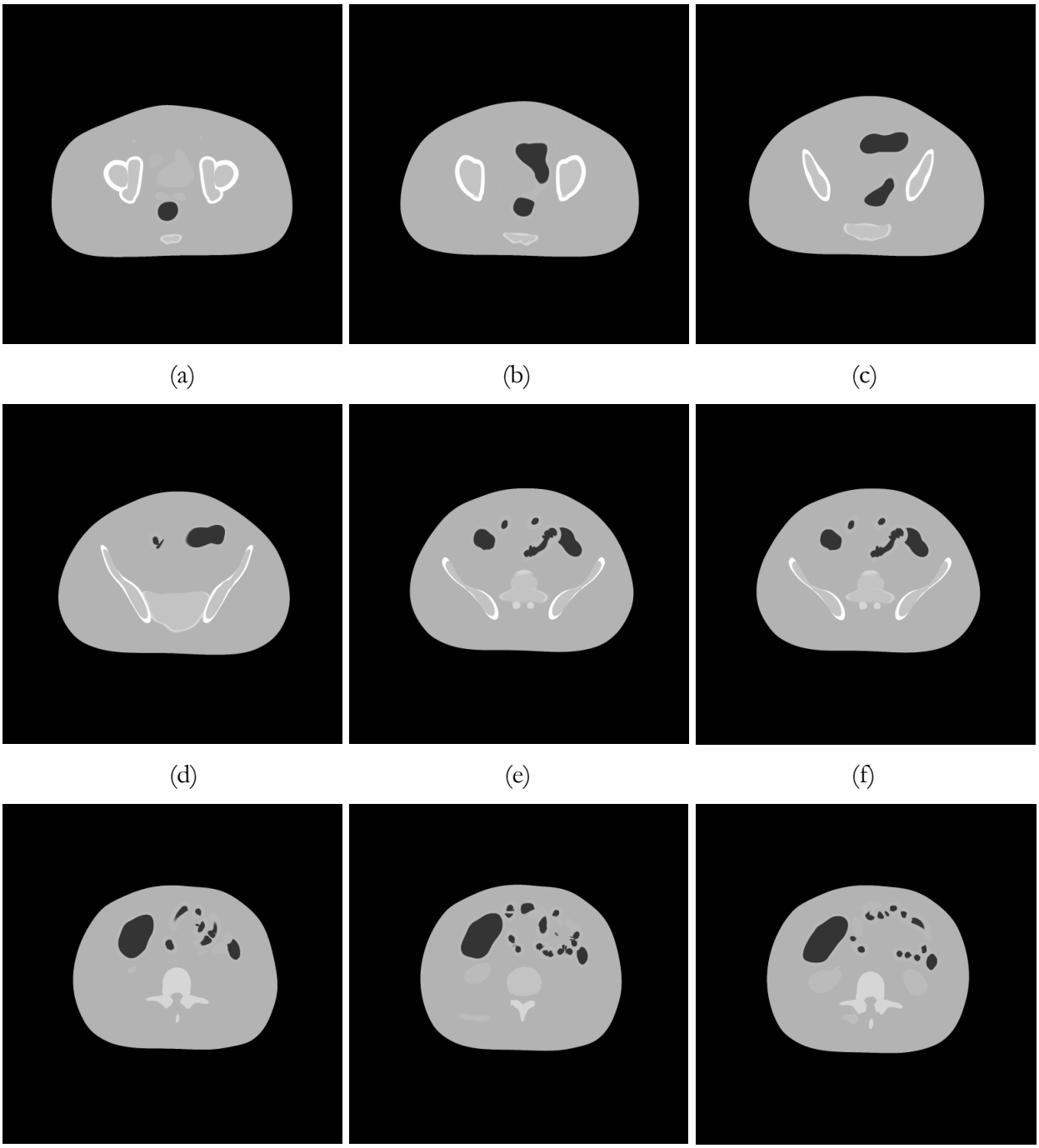


Figure 2.6: The NCAT phantom for (a) The 15th slice (b) The 26th slice (c) The 41th slice (d) The 57th slice (e) The 81th slice (f) The 102th slice (g) The 123th slice (h) The 134th slice (i) The 147th slice. Display window: [0,0.0176] mm^{-1}.

2.4 GPU Acceleration

Based on the Helical CT Advanced Reconstruction Engine (HECTARE) software package by Dr. Daniel Keesing, Ayan Mitra for the laboratory of Dr. Joseph O'Sullivan generated parallel accelerated code in multi-GPU systems using CUDA programming tools. The GPU-accelerated code achieved a speedup of 72 times over the HECTARE.

2.4.1 Hardware

In this work, we start with an Intel Xeon E5 2630-v4 consisting of 10 cores and 20 threads and an installed memory of 128 Gigabytes. The base frequency of the E5 2630-v4 is 2.2 GHz. We use four GeForce GTX 1080Ti for GPU computation. The GeForce GTX 1080Ti is based on Pascal architecture with 28 multiprocessors, 128 CUDA cores, 3584 cores, a 1582 MHz boost clock and 11172 Mbytes of global memory. Each block contains 65536 registers and 49152 bytes of shared memory, with a maximum of 1024 threads.

2.4.2 CUDA

CUDA is a parallel computing platform developed by NVIDIA, which supports a majority of programming languages such as C, C++, Java, Python, etc. For a CUDA kernel, we could launch at most 65535×65535×65535 blocks, and each block has a maximum number of 1024 threads. In this case, theoretically, one GPU could run up to 65535×65535×65535×1024 concurrent processes. However, the maximum number of threads is also restricted by the features of the GPU.

Figure 2.7 shows the computations processed in GPUs, including backprojection, exponentiation, projection and updating. In projection and backprojection computation, we set the number of threads per block to 256 due to the great amount of resources used per block. The size of a grid is 6,960 and the maximum number of 1,781,760 threads run simultaneously.

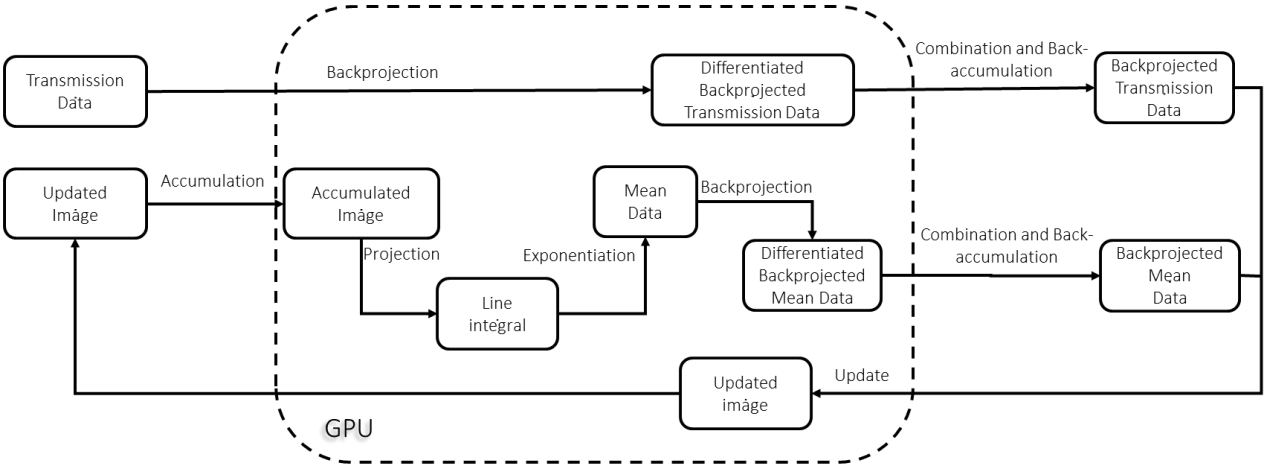


Figure 2.7: The computations in GPU

Chapter 3

Acceleration

With GPU acceleration, the run time of the AM algorithm for CT reconstruction is 19.6 seconds per iteration. In other words, running the AM algorithm over 200 iterations takes approximately 1 hour. In the current clinical environment, doctors and researchers could not wait for such a long time to see the result. Therefore, other acceleration methods must be implemented to reduce the computation time.

3.1 Feldkamp-Davis-Kress (FDK) as the Initial Condition

3.1.1 Introduction to the FDK Algorithm

The FDK algorithm is a three-dimensional standard filtered backprojection algorithm. In this thesis, we use the helical FDK algorithm in the rebinned geometry introduced by Tang et al [6].

The FDK algorithm contains several data preprocessing operations and a backprojection procedure.
1. Linearly interpolate the fan-beam data to perform row-wise fan-beam-to-parallel-beam rebinning
2. Apply cosine weight to deal with the divergence of the x-ray source in the η direction. The cosine weight is defined as

$$R_f / \sqrt{{R_f}^2 + v^2}$$

3. Apply the ramp filter. In our implementation, Hann window is used: $0.5 + 0.5\cos(\pi\omega)$.
4. Use a redundancy weight to normalize the contribution of different views to each voxel. The 3-D weighting function is

$$w_{3D}(\theta,t,v)=\frac{w_{2D}(\theta,t)|v_c|^2}{w_{2D}(\theta,t)|v_c|^2+w_{2D}(\theta_c,t_c)|v|^2},$$

where the subscript c refers to the complementary ray, t is the detector position along the γ direction, while v is the detector position along the η direction, θ is the angle between the x axis and line connecting the source and fan position $(\beta=\gamma+\theta)$, and

$$w_{2D}(\theta,t)=\begin{cases}1+\dfrac{\theta}{\pi} & if\ -\pi\le\theta<0,\\ 1-\dfrac{\theta}{\pi} & if\quad 0\le\theta<\pi.\end{cases}$$

5. The overall backprojection expression is

$$\mu(x,y,z)=\frac{1}{2}\int_{-\pi}^{\pi}\frac{R_f}{\sqrt{R_f^2+\eta^2}}\mathrm{w}_{3D}(\theta,t,v)\rho(\theta,t,v)d\theta,$$

where (x,y,z) refers to a voxel in the reconstructed image, and $\rho(\theta,t,\eta)$ is the filtered projection data.

3.1.2 Result

To assess the acceleration performance of the FDK image as the initial condition, we ran AM algorithm with different initial conditions (zeros and the FDK image) and plotted the objective function versus the number of iterations. Figure 3.1 is a comparison between an AM algorithm image initiated with zeros and with FDK image for the clinical data.

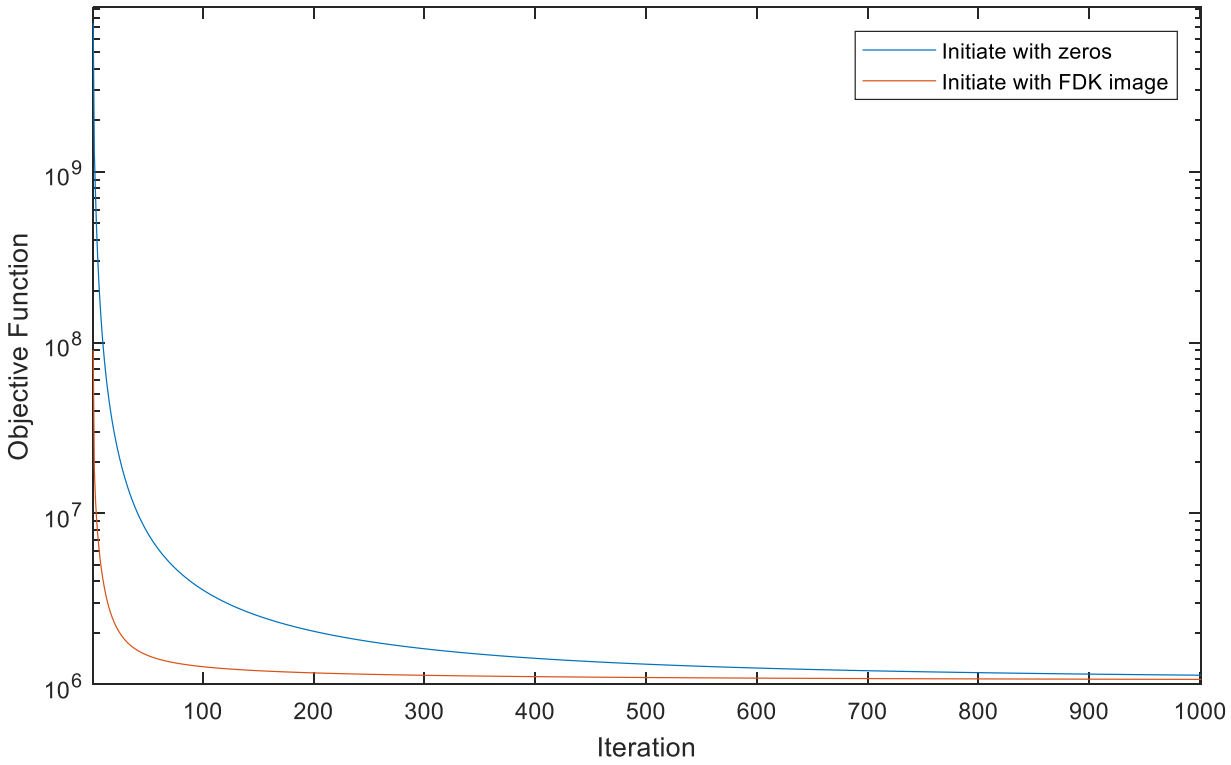


Figure 3.1: Plot of objective function of AM images computed with different initial conditions

The objective function using the FDK image as the initial condition is equivalent to the objective function of the 8th iteration output with zero initial condition. However, the objective function of the 300th iteration with the FDK initial condition is approximately the same as the objective function of the 1057th iteration with zero initial condition. Therefore, the FDK image, as the initial image, sped up convergence of AM algorithm not only by decreasing the initial value, but also by allowing for larger decreases of objective function as iterations proceed toward the convergence. In other words, the FDK image provides us a shortcut towards the optimal solution. For this research environment, including the processing unit and transmission data we used, since the run time for FDK algorithm is approximately 8 seconds, it is worth using the FDK image to triple the convergence speed.

3.2 Ordered subsets (OS)

3.2.1 Introduction

The ordered subsets method is an acceleration method for the convergence of iterative algorithms. The idea is to group the projection data into an ordered sequence and process the data sequence in subiterations.

Given a fixed number of ordered subsets N, we divide the projection data into N ordered subsets; in most cases, therse subsets do not overlap and therefore from a paritition of the total projection data. It is common to select subsets that are balanced, each having the same amount of projection data. We will only consider such balanced, non-overlapping subsets. Since we have exploited quarter-rotation symmetry, the number of ordered subsets should be a factor of the number of views per quarter rotation (290 for the Siemens geometry), which is one of the following numbers: 2, 5, 10, 29, 58, or 145. When processing the N^{th} ordered subset, the image is updated based on the N^{th} subset of projection data and the $(n-1)^{\text{st}}$ result. Since during the processing of each subsets, only $1/N$ of the data is used, the scalar for the penalty term must also be adjusted to be λ/N.

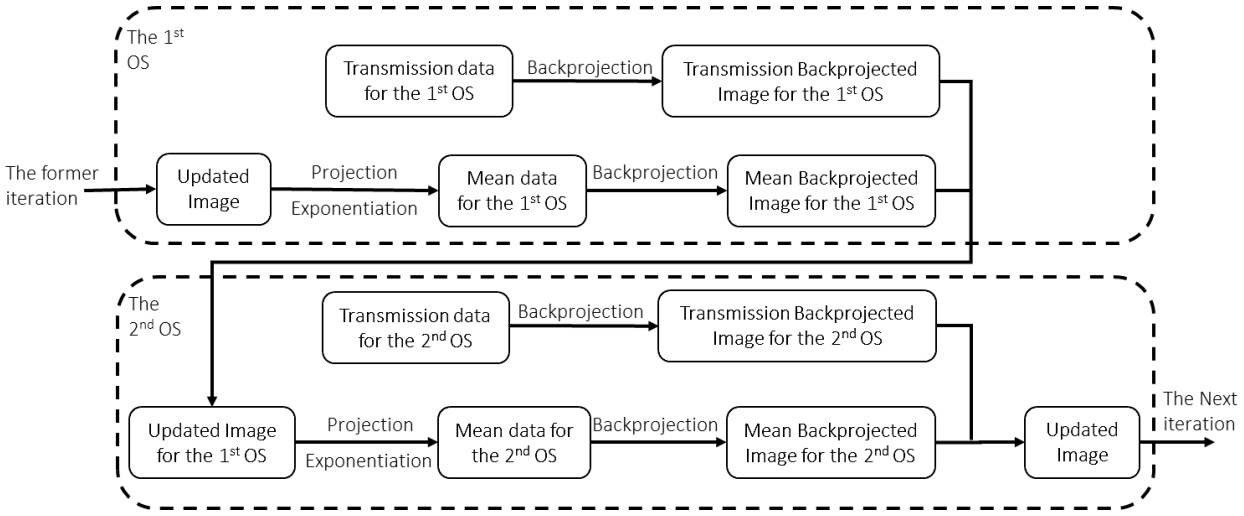


Figure 3.2: An example of an ordered-subsets AM algorithm with 2 OS

Since the projection data used in every-ordered-subset computation is $1/N$ of the entire data, the computing cost of projection and backprojection for each ordered subset is also $1/N$ of the cost for the entire date set, and the total cost of projection and backprojection is approximately the same as for the algorithm without ordered subset. In many cases, convergence is accelerated by the number of ordered subsets N. Theoretically, a reconstruction algorithm with N subsets converges N times faster than the algorithm without ordered subsets, at least for the initial iterations.

However, with the ordered-subsets method, the AM algorithm is not guaranteed to converge. The result may not be an optimal solution. Ahn, Fessler et al. [7] proposed a convergent ordered-subsets algorithm which is guaranteed to converge with any surrogate function that meets their requirement. Moreover, the ordered-subsets-switching method could also be utilized for seeking the optimal solution.

3.2.2 Analysis of Convergence Based on Number of Ordered Subsets

In order to find the optimal number of ordered subsets, unregularized AM algorithms with different numbers of ordered subsets initiated with the an FDK image have been run on the clinical transmission data. Figure 3.3 shows the objective function versus time for different numbers of OS.

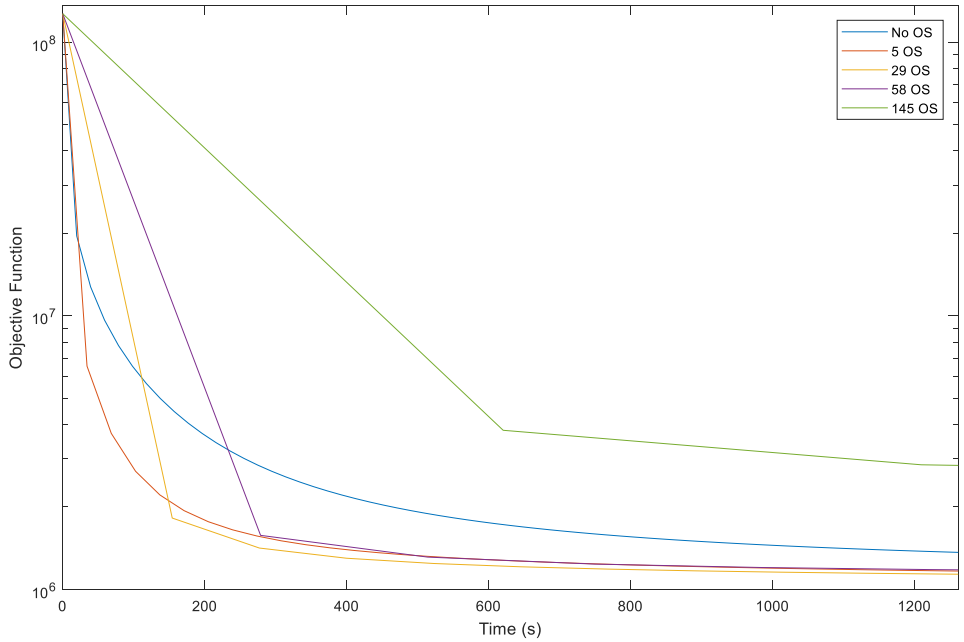


Figure 3.3: Plot of objective function versus different number of ordered subsets

From Figure 3.3, the plot of the objective function of the algorithm with 145 OS is far away above the others, which means that the algorithm with the most ordered subsets does not give the best performance for this data set. The algorithm with 29 ordered subsets always performs the best from 200 to 1200 seconds.

According to Chapter 3.2.1, the acceleration rate of the OS method should be approximately equal to the number of subsets. We can use this principle to judge if our algorithm with OS reaches the limit of the OS method. Figure 3.4 shows the relationship between the value of the objective function with different numbers of ordered subsets and without ordered subsets. The y-axis stands for the no-OS-equivalent iteration, which is the number of iterations without ordered subsets that the AM algorithm with the specific number of OS could achieve from the $(i-1)^{\mathrm{st}}$ iteration to the i^{th} iteration. For example, in Figure (b), the 1st point means that the objective function of the 1st iteration of the AM algorithm with 29 ordered subsets approximately equals the objective function of the 28th iteration of the AM algorithm without ordered subsets, and the 2nd point means that the decrease of objective function from the 1st to the 2nd iteration requires the AM algorithm without OS to run over 27 additional iterations. The acceleration rate is given by

$$\text{Acceleration Rate}\,(s,n) = \text{iter}_n^s - \text{iter}_{n-1}^s$$
$$\text{iter}_n^s = \text{argmin}_{\text{iter}}|\text{obj}^1(\text{iter}) - \text{obj}^s(n)|,$$

where s is the number of ordered subsets, n is the number of iterations, $\text{obj}^s(n)$ denotes the objective value with s ordered subsets at the n^{th} iteration. We use this no-OS-equivalent iteration as the acceleration rate of the OS method.

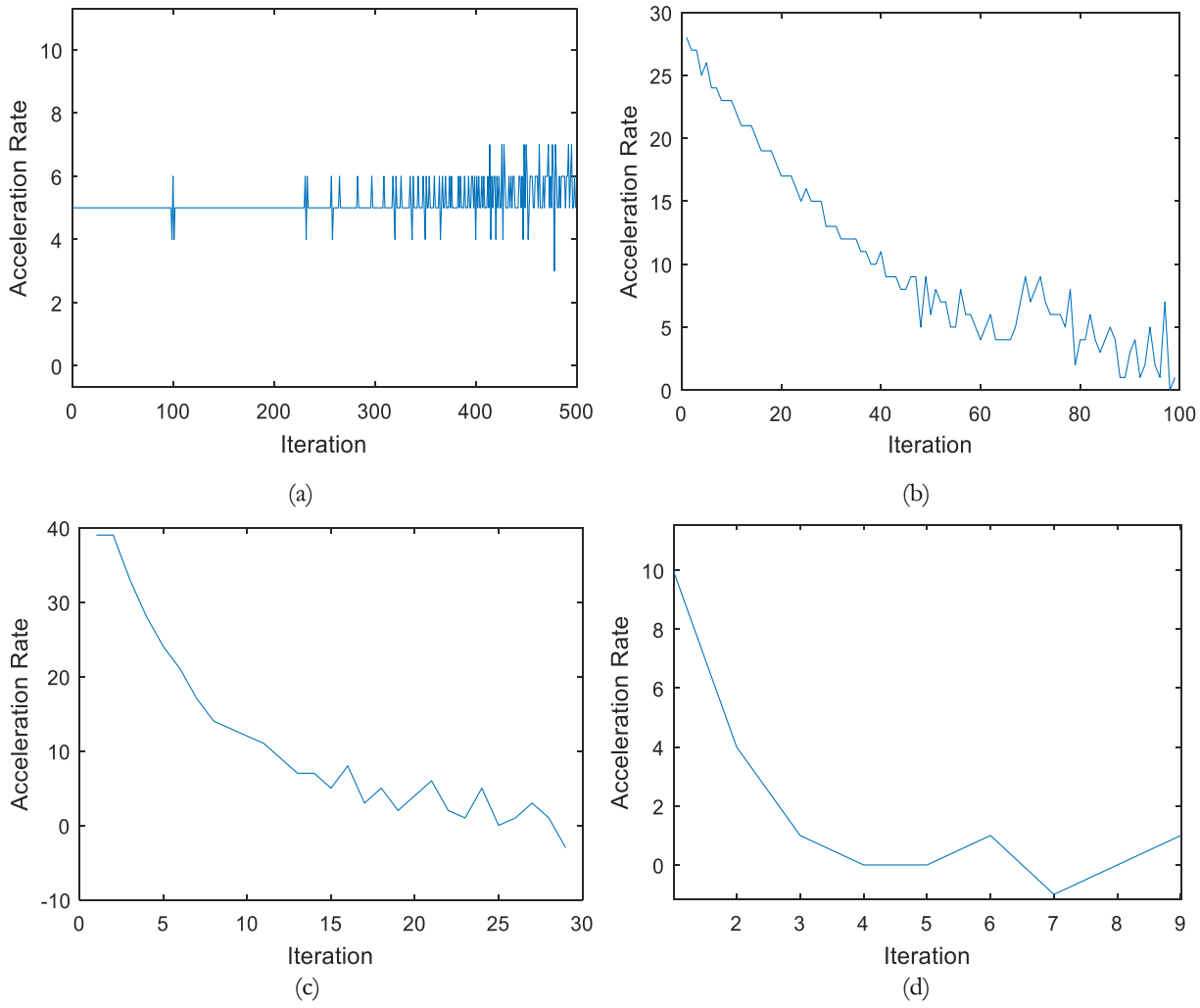


Figure 3.4: Acceleration rate of OS method with different numbers of OS (a) 5 ordered subsets, (b) 29 ordered subsets, (c) 58 ordered subsets, and (d) 145 ordered subsets

Figure 3.4 provides a new way to assess the performance of different numbers of ordered subsets. The acceleration rate of 29 ordered subsets starts at 28, and drops to 5 in 60 iterations. The acceleration rate of 58 ordered subsets starts at 39, swiftly dropping to 5 in 13 iterations. The acceleration rate of 145 ordered subsets starts at 10, which is much smaller than our expectation. However, the acceleration rate of 5 OS remains nearly the same for 500 iterations. To summarize, 145 OS already reached the limit at the 1st iteration, the accelerating rate of 58 OS dropped quickly, 29 OS exhibited a good performance at the start, and 5 OS was mostly stable for 500 iterations. In other words, 145 OS could never be utilized in our implementation, regardless of the run time.

3.2.3 Results

In Section 3.2.2, 29 was shown to be the best number of ordered subsets for a 20 minutes' computations. Figure 3.5 shows a comparison of the regularized objective function between 29-ordered-subsets and no-ordered-subset algorithms.

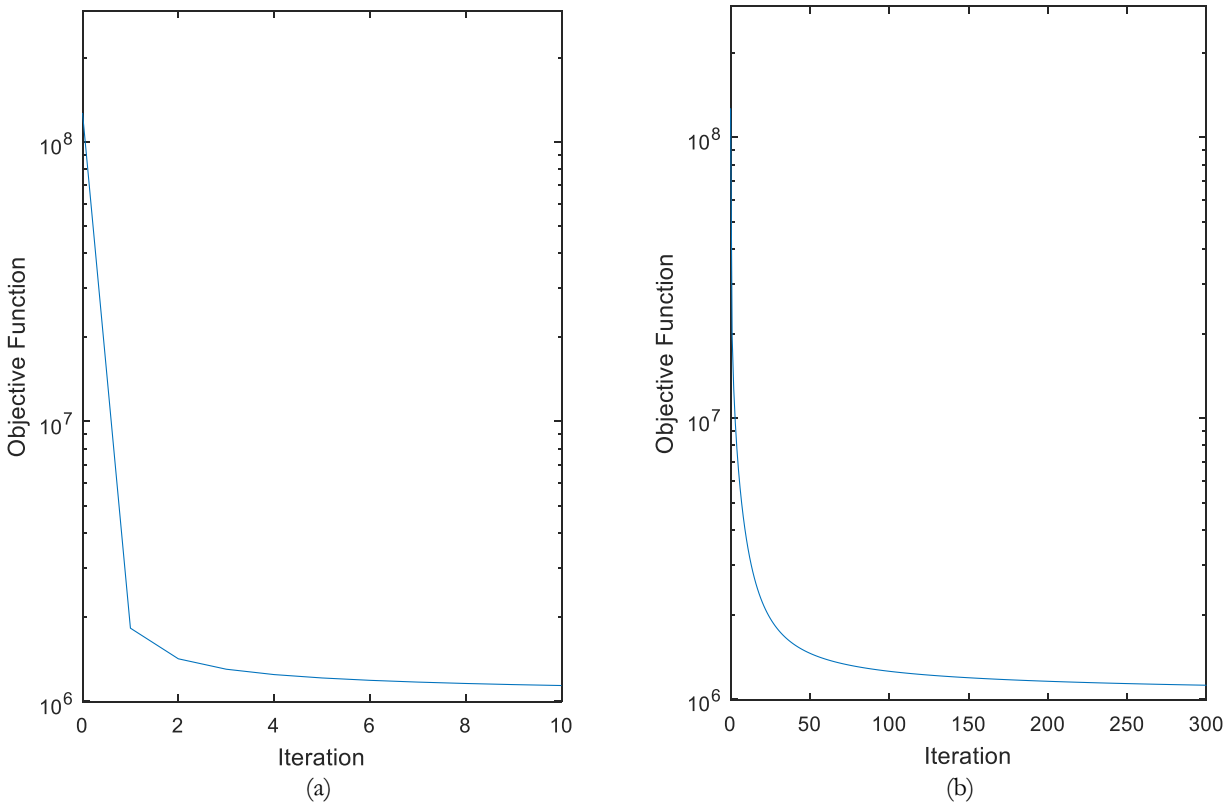
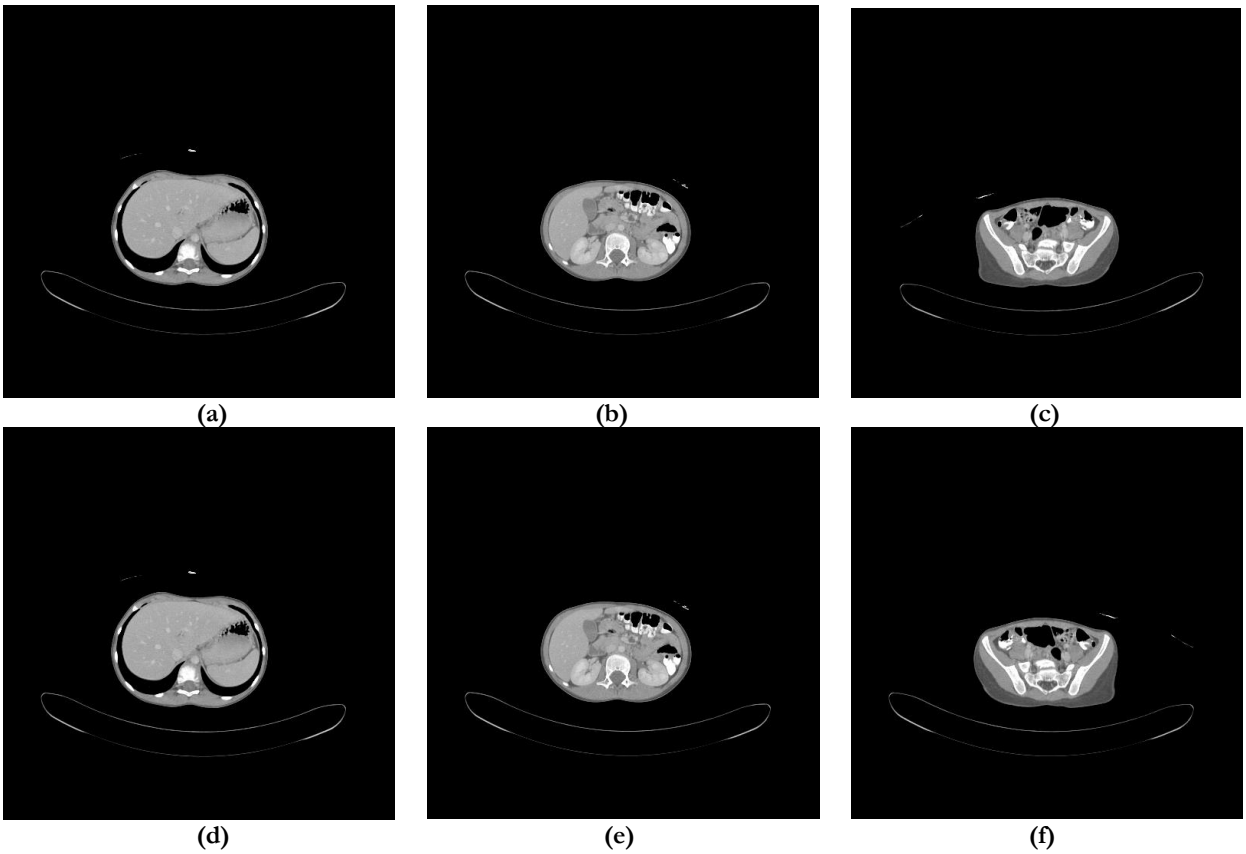


Figure 3.5: Log of objective function with different numbers of ordered subsets. (a) 29 ordered subsets with FDK initial condition. (b) No ordered subsets with FDK initial condition.

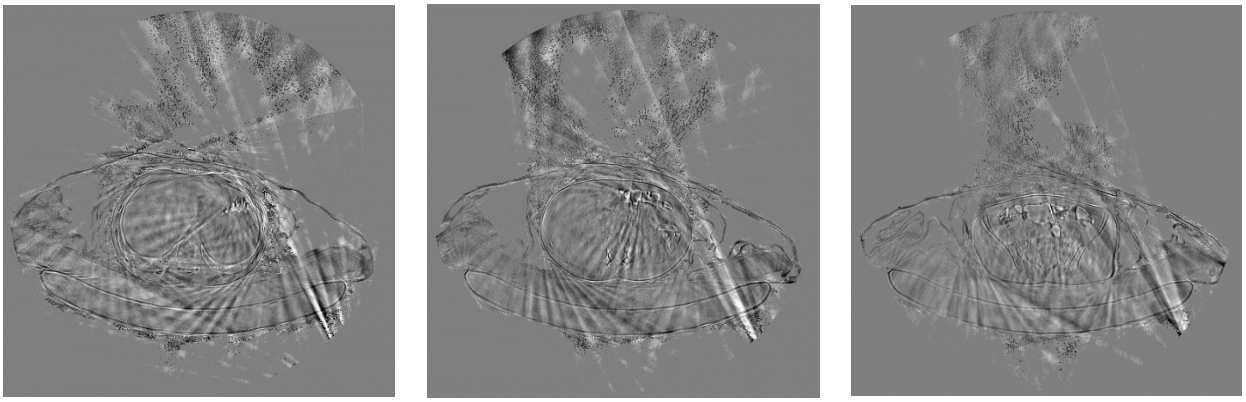
The objective function of images produced by the AM algorithm with 29 ordered subsets at the 10th iteration is about the same as the objective function of the AM algorithm without ordered subsets at the 251st iteration. The acceleration rate with respect to number of iterations required to achieve a specific value of the objective function is about 25 in this example.

It has been shown that the OS method great accelerates our reconstruction algorithm. However, images may dramatically differ from each other even if they have approximately the same values of the objective functions. We want to achieve the same solution using the OS method as from the non-OS method. In order to assess whether 29 ordered subsets over 10 iterations is equivalent to no ordered subsets over 251 iterations, we computed the difference between the two reconstructed images, as shown in Figure 3.6. The worst-case percentage difference is approximately 0.25%. Therefore, the ordered-subsets algorithm achieved a similar result as the standard AM algorithm.



(a) **(b)** **(c)**

(d) **(e)** **(f)**



(g) **(h)** **(i)**

Figure 3.6: Comparison between the result of the AM algorithm with 29 OS at the 10th iteration and the result of the AM algorithm without OS at the 251st iteration. (a) The 48th slice of non-OS-251-iter image, (b) The 77th slice of non-OS-251-iter image, (c) The 126th slice of non-OS-251-iter image, (d) , The 48th slice of 29-OS-10-iter image, (e) The 77th slice of 29-OS-10-iter image, (f) The 126th slice of 29-OS-10-iter image, (g) The 48th slice of the difference between the 29-OS-10-iter image and the non-OS-251-iter image, (h) The 77th slice of the difference between 29-OS-10-iter image and the non-OS-251-iter image, (i) The 126th slice of the difference between the 29-OS-10-iter image and the non-OS-251-iter image. The display windows for (a), (b), (c), (d), (e) and (f) are each $[0.0129,0.0259]$ mm^{-1}. The display windows for (g), (h), (i) are each $[-5 \times 10^{-5}, 5 \times 10^{-5}]$ mm^{-1}.

3.3 Code-Based Acceleration

Our goal of acceleration is to reduce the necessary run time of the algorithm. Although we have successfully found the optimal number of OS and reached the acceleration rate of 25 with respect to iterations for the example in Section 3.2, the run times of the AM algorithm with no ordered subsets for 251 iterations and 29 ordered subsets for 10 iterations are 4769 seconds and 1273 seconds, respectively. That means the acceleration rate with respect to time is just about 4. As mentioned in Chapter 3.2.1, the time required to compute projections and backprojections does not change much with different numbers of ordered subsets, but the algorithm with 29 OS is 6.7 times slower than the algorithm without ordered subsets. In this section, we will discuss the run time of the code and provide some solutions.

3.3.1 Run Time Analysis

To get the detailed runtime of our code, Visual Studio time analysis and the NVIDIA CUDA profiler were combined to generate Table 3.1.

Table 3.1: Run time of different processes versus different numbers of ordered subsets

Parts	Processes	No ordered subset			5 ordered subsets			29 ordered subsets		
		Total time(s)	Percentage	Num	Total time(s)	Percentage	Num	Total time(s)	Percentage	Num
CPU part	Config reading	0.453	1.19%	1	0.448	0.90%	1	0.447	0.35%	1
	Geometry computing	0.828	2.17%	1	0.816	1.63%	1	0.827	0.65%	1
	Data reading	0.738	1.93%	1	0.734	1.47%	1	0.726	0.57%	1
	Parameter allocating	0.825	2.16%	1	0.783	1.57%	1	0.780	0.61%	1
	Combination	0.306	0.80%	1	1.455	2.91%	5	6.177	4.87%	29
	Accumulation	0.845	2.21%	1	4.375	8.76%	5	24.766	19.52%	29
	Backprojected data output	0.863	2.26%	1	4.100	8.21%	5	22.838	18.00%	29
	Combination	0.245	0.64%	1	1.080	2.16%	5	6.873	5.42%	29
	Accumulation	0.838	2.20%	1	4.160	8.33%	5	23.113	18.21%	29
	Pre-updating	0.497	1.30%	1	1.295	2.59%	5	5.533	4.36%	29
	Result output	1.019	2.67%	1	1.154	2.31%	1	0.993	0.78%	1
	CPU total	7.457	19.53%		20.400	40.85%		93.073	73.35%	
GPU part	MemCpy HtoD	0.286	0.75%	27	0.920	1.84%	51	4.892	3.86%	243
	MemCpy DtoH	0.225	0.59%	9	0.521	1.04%	25	2.371	1.87%	145
	Back Projection	25.030	65.57%	10	24.969	50.00%	10	23.136	18.23%	58
	Forward Projection	3.080	8.07%	5	3.095	6.20%	5	3.249	2.56%	29
	Exponentiation	0.006	0.02%	1	0.005	0.01%	5	0.006	0.00%	29
	update	0.005	0.01%	1	0.028	0.06%	5	0.164	0.13%	29
	GPU total	28.632	75.00%		29.538	59.15%		33.818	26.65%	
Total	Total	38.175			49.938			126.891		

The run times of GPUs are about the same regardless of the number of subsets, which means using ordered subsets does not influence the time of the GPU computation significantly, which was as expected. However, with the increase of the number of OS, the run times of the CPU part increases a lot, especially in "Combination" and "Accumulation". For every ordered subset, the combination and accumulation processes always deal with the whole backprojected image with size of 512×512×164, thus the run time per iteration of these two parts is proportional to the number of ordered subsets. As a result, the run time of the CPU computations with 29 ordered subsets is approximately 12.5 times greater than the run time of the CPU computation without ordered subsets. In other words, as the number of subsets increases, we are losing our advantage of GPU acceleration.

In order to reduce the run time of combination and accumulation, we generated GPU-accelerated combination and accumulation code. In our GPU-accelerated version, we mainly use the a Hillis-Steele scan to parallelize the accumulation procedure and use shared memory in CUDA to speed up the read and write processes.

3.3.2 Hillis-Steele Scan

The Hillis-Steele scan, derived by Daniel Hillis and Guy L. Steele [6], is a decoupled inclusive sum method, with $O(\log N)$ steps and work $O(N \log N)$, which means the run time of the accumulation process could be reduced from N to $\log N$, theoretically.

A Hillis-Steele scan has $\text{ceil}(\log_2 N)$ iterations. For every iteration j, the i^{th} element is added to the $(i-j)^{\text{th}}$element. Figure 3.7 shows the process of the Hillis-Steele scan of 7 elements.

Since the GPU version of accumulation process simultaneously deals with 512 images of size 512×164, the number of run steps is $\text{ceil}(\log_2(512) + \log_2(164)) = 17$, instead of 512×164.

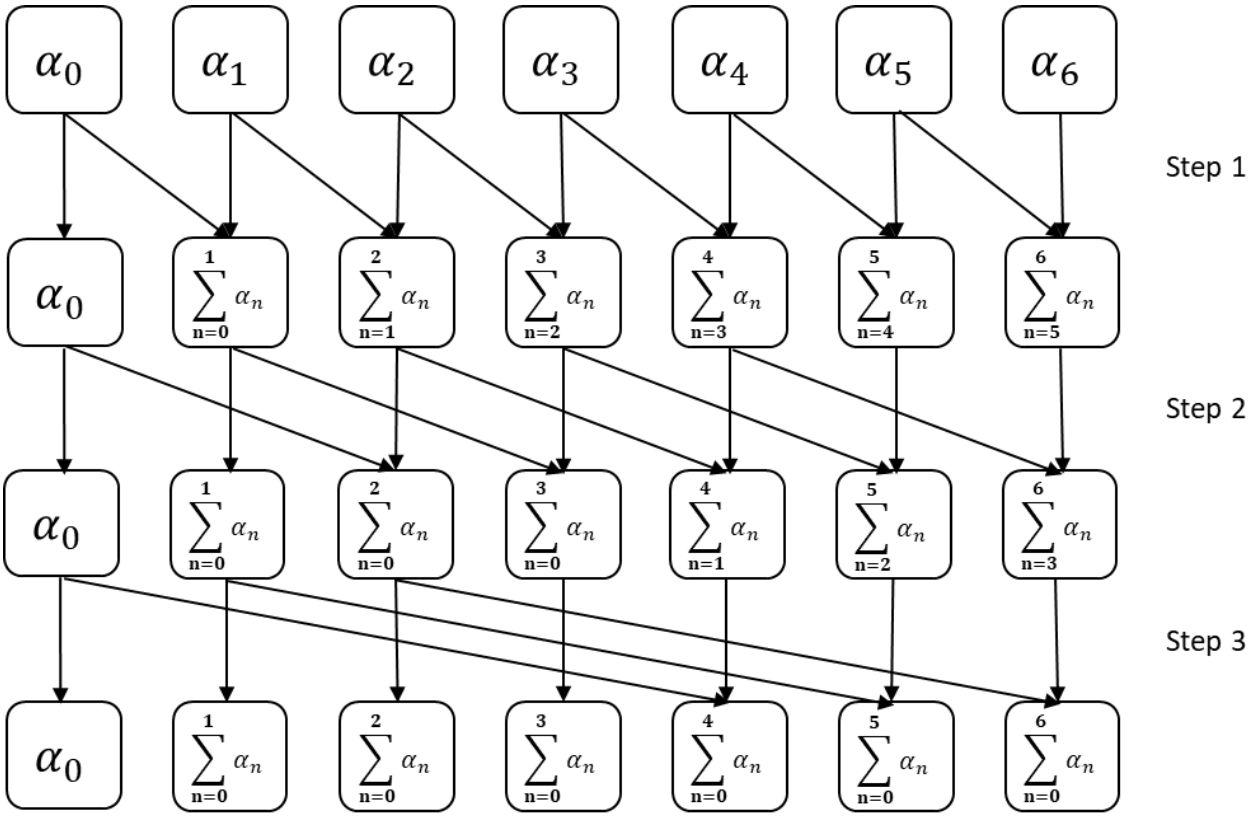


Figure 3.7: Process of Hillis-Steele scan of 7 elements

3.3.3 Shared Memory

Shared memory in CUDA is an on-chip memory that can be accessed concurrently by all the threads in the same block without any conflict. Compared to global memory, shared memory is much faster and easier to synchronize. Since in every iteration in a Hillis-Steele scan, the summation process always deals with the same data, we can read the data from global memory into shared memory and write it to global memory after the scanning iterations. The number of threads per block is set to 512 to match the size of the data to be accumulated. A block in NVIDIA GTX 1080 Ti has 49,152 bytes of shared memory, which is enough for our Hillis-Steele scan.

3.3.4 Results

The run time of the combination and accumulation processes decreases from 1.13 seconds to 0.37 seconds. Therefore, this modification saved about 0.8 second/subset/iteration. Taking 29 ordered subsets, for instance, our GPU implementation of the AM algorithm saves 23.4 seconds per iteration.

Table 3.2 Run time comparison between original code and modified code with different OS

	No Ordered Subsets		5 Ordered Subsets		29 Ordered Subsets	
Procedure	Original	Modified	Original	Modified	Original	Modified
Combination	0.306	0.084	1.455	0.395	6.177	1.885
Accumulation	0.845	0.201	4.375	0.92	24.766	5.278

Since the AM algorithm with more OS takes greater advantage of this code modification, Figure 3.8 is plotted to show that 29 ordered subsets still perform the best among all the numbers of ordered subsets in 20 minutes for the clinical data. Figure 3.9 compares the projected result of the modified code and the original code for the clinical data. The worst-case percentage error is approximately 6×10^{-6} %.

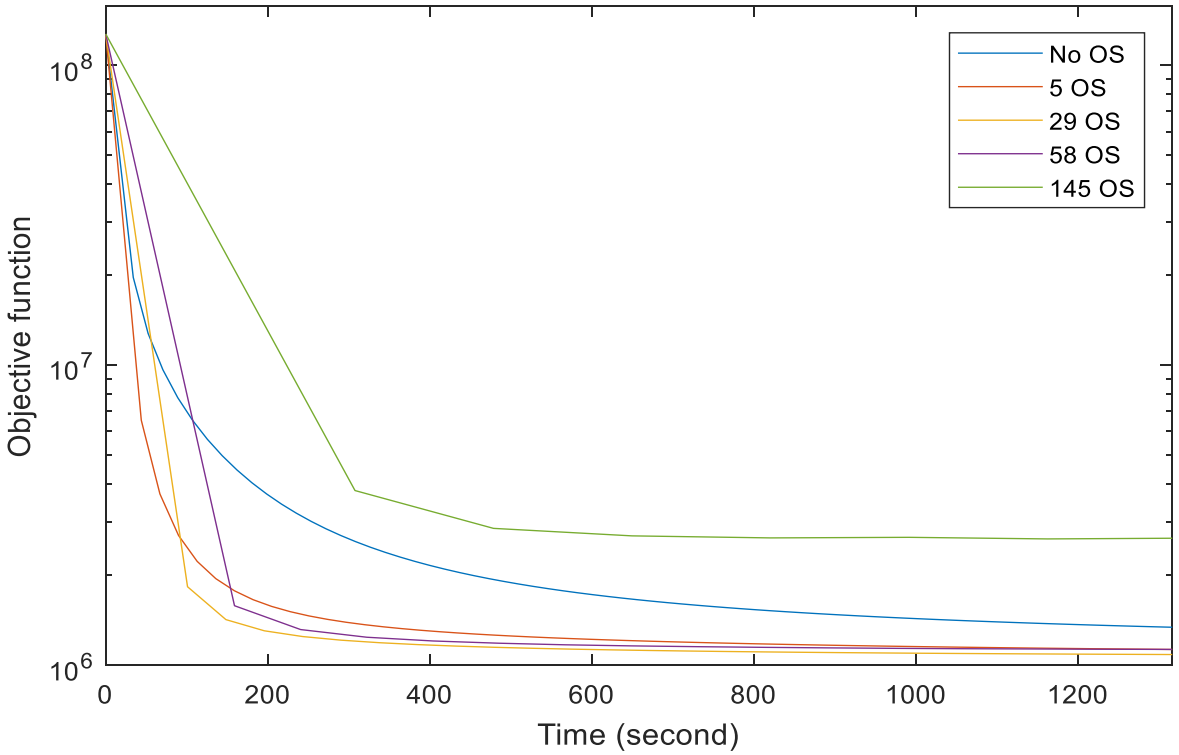
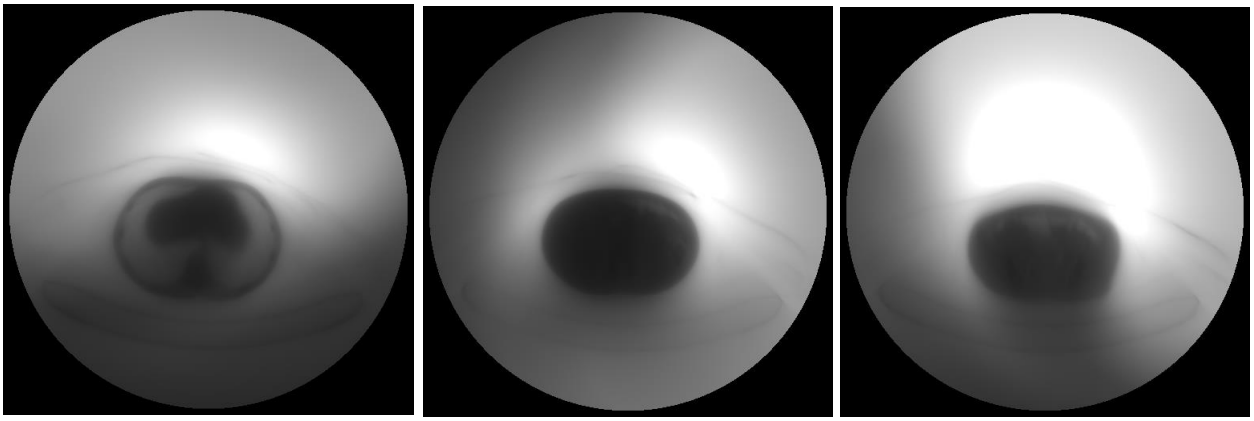


Figure 3.8: Plot of objective function versus different number of ordered subsets of modified code



(a) **(b)** **(c)**

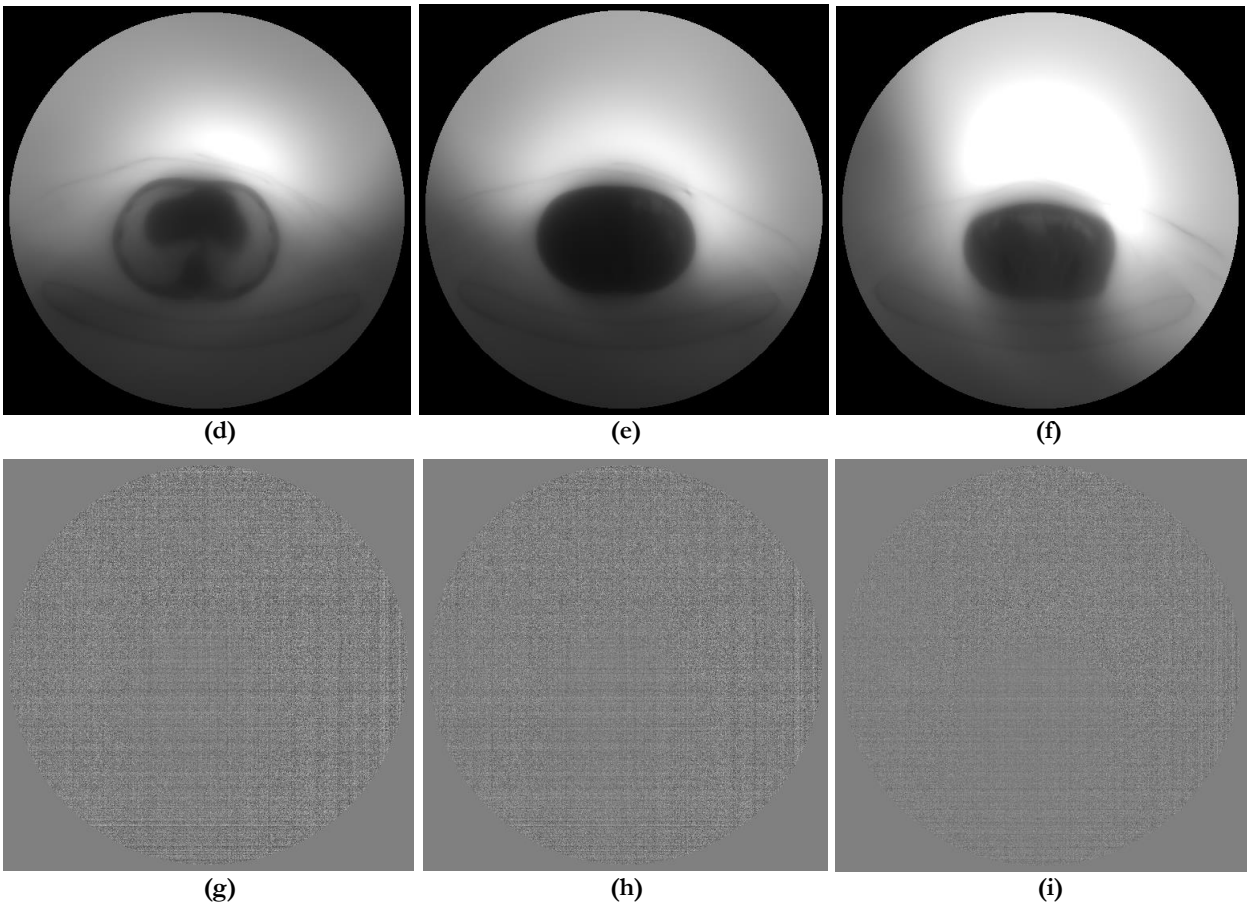


Figure 3.9 Comparison of backprojected images between the results of the original implementation and the modified implementation. (a), (b), and (c) are the results of the modified implementation. (d), (e), and (f) are the results of the original implementation. (g), (h), and (i) are the differences. (a), (d), and (g) are the 38th slice. (b), (e), and (g) are the 75th slice. (c), (f), and (i) are the 124th slice. The display windows for (a), (b), (c), (d), (e), and (f) are each $[0,168595]$ mm^{-1}. The display windows for (g), (h), and (i) are each $[-0.0,0.01]$ mm^{-1}.

3.4 Hybrid Ordered-subsets Method

According to a convergence analysis, the AM algorithm with ordered subsets is not guaranteed to converge. As a result, the number of ordered subsets should be switched (reduced) after a specified number of iterations in order to reach the global optimum.

To ascertain the switching point, Figure 3.10 shows a plot of the gradient of the objective value versus the value of the objective function with the different numbers of ordered subsets. The gradient of the

objective value measures the descent per second of our algorithm with a specific number of ordered subsets at a specific point, which is computed as $\left(\frac{\text{current objective value}-\text{next objective value}}{\text{run time per iteration}}\right)$, based on the assumption that images with the same objective value generated by AM algorithms with different ordered subsets are approximately the same.

If the gradient of the objective value for other numbers of ordered subsets is greater than the gradient of the objective value for the current number of subsets, the number of subsets should be changed. From Figure 3.10, the gradient of the objective function of 29 OS crosses the gradient objective function of 5 OS at the objective value of 1.074×10^6, which is the 45th iteration of AM algorithm with 29 OS. Then, approximately 900 seconds would be saved.

However, in a clinical environment, the expected run time of our reconstruction implementation is at most 20 minutes, while the hybrid-ordered-subsets method is available with the reconstruction running for more than 2185 seconds. This method may be utilized in a more time-insensitive task.

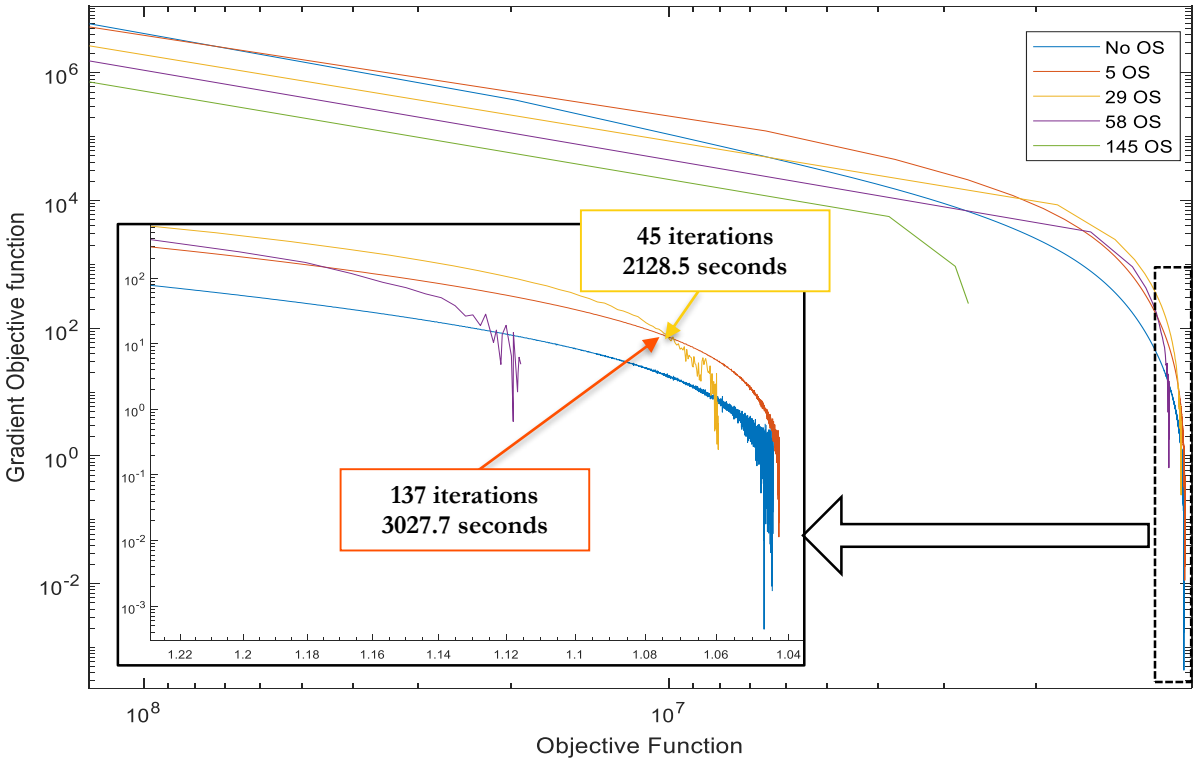


Figure 3.10: Gradient of the objective value versus the value of objective function with different numbers of ordered subsets

Chapter 4

Code-related Problems

During the optimization process, several problems related to our implementation were found. This chapter presents the analysis of these code-related problems and discusses the explanation and solutions.

4.1 Problem Descriptions

4.1.1 Increasing-objective-value Problem

The objective-function-increasing problem was first discovered when we compared the performance of our algorithm with different numbers of ordered subsets. With FDK image as the initial condition, the objective function decreased initially, bounced up at 3500th iteration and increased with the rate of 20/iteration eventually, which is shown in Figure 4.1.

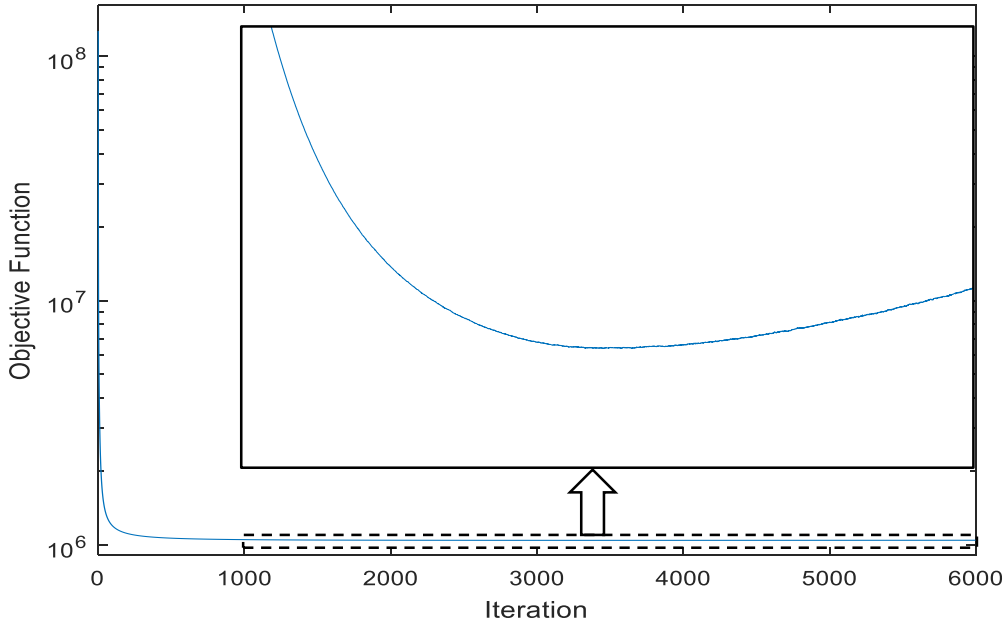


Figure 4.1: The objective function of the regularized AM algorithm starting from FDK image without ordered subsets with $\boldsymbol{\lambda}$=100, $\boldsymbol{\delta}$=0.0002

Increase of the objective function is a severe problem, since it may indicate the failure of convergence of our implementation. In order to analyze the convergent property of the updating algorithm, we came up with the following assumptions and set up corresponding experiments.

Inappropriate penalty term: Since the penalty term from Lange [9] is adopted in our objective function, we want to see if inappropriate regularization parameters influence the convergence properties of the AM algorithm. To assess the influence of the penalty term, Figure 4.2 shows the plot of the objective function of unregularized AM algorithm without ordered subsets initiated with the FDK image and computed for 6000 iterations. It is shown that the weight of the penalty term did not change the trend of the curve. The objective function without a penalty, which is just the I-divergence, also increased after some specific number of iterations. Therefore, the penalty is not the reason for the increasing objective function.

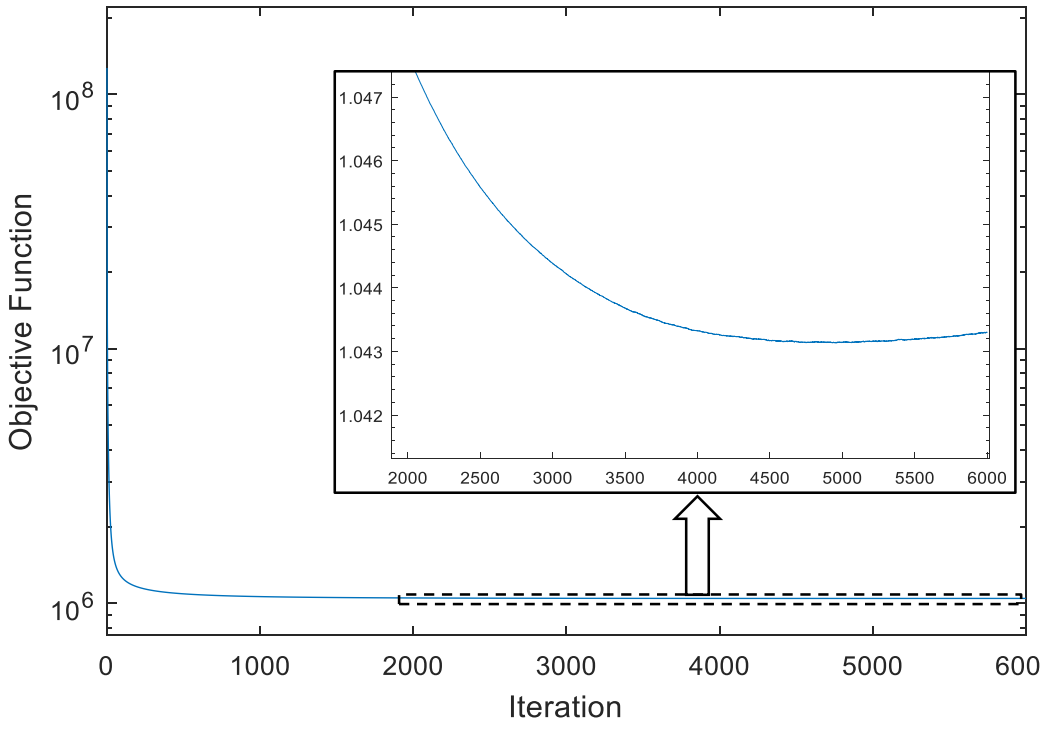


Figure 4.2: The objective function of the unregularized AM algorithm

Value vibration: We observed that the objective function of the AM algorithm increases after 3500 iterations. We want to know if the objective function will keep increasing, or it will bounce up and down. The objective function of the AM algorithm for 20,000 iterations is plotted in Figure 4.3. The objective function keeps increasing from the 3,500th iteration to the 20,000th iteration.

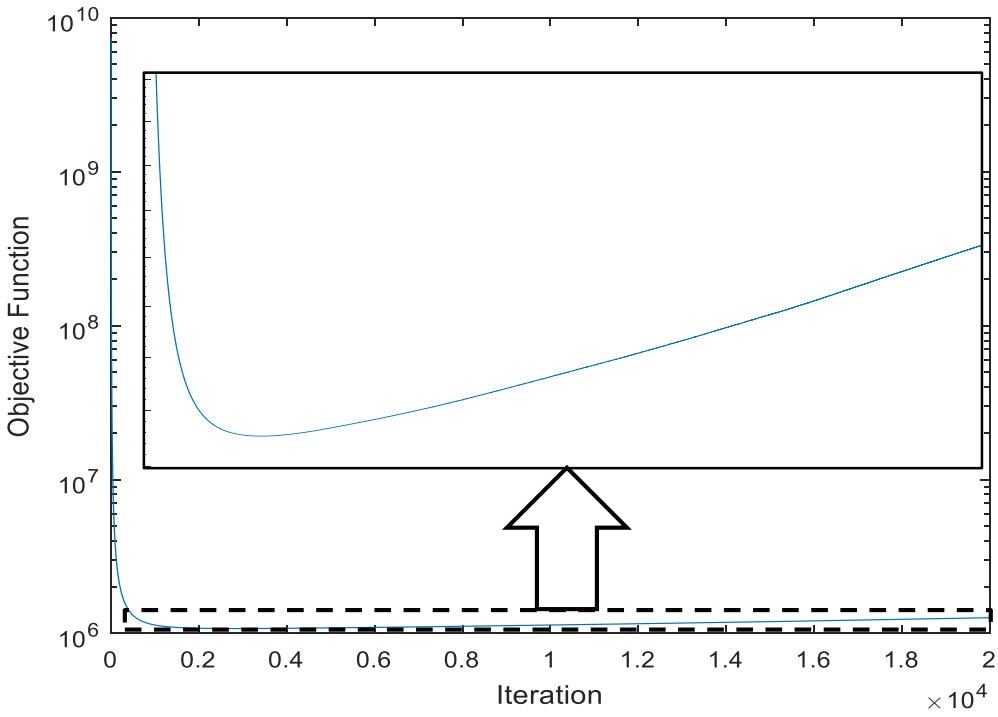


Figure 4.3: The objective function of the unregularized AM algorithm over 20,000 iterations starting at zeros

The problem caused by the GPU implementation: The AM algorithm and the branchless distance-driven method was first implemented on CPUs by Dr. Daniel B. Keesing [5], in a software package he called HECTARE. HECTARE code was accelerated by Ayan Mitra on GPUs [4]. We want to compare the output of the GPU code to the CPU code, to see whether the CPU code has the same problem. Since the run time of the CPU code is 300 seconds per iteration, it is unrealistic to run it for over 20,000 iterations. An alternative experiment is required. We ran the CPU version of the AM algorithm with the output image of the AM algorithm after 20,000 iterations (computed on GPUs) as the initial condition. If the CPU code was not suffering the same problem, the objective function would have a decreasing trend. Figure 4.4 is a plot of the objective function of the CPU-based AM algorithm starting at the image after 20,000 iterations. The objective function of the CPU code is also

increasing, which shows that the original CPU implementation also encounters the problem of increasing objective function value, but the plotted values jitter more than for the GPU.

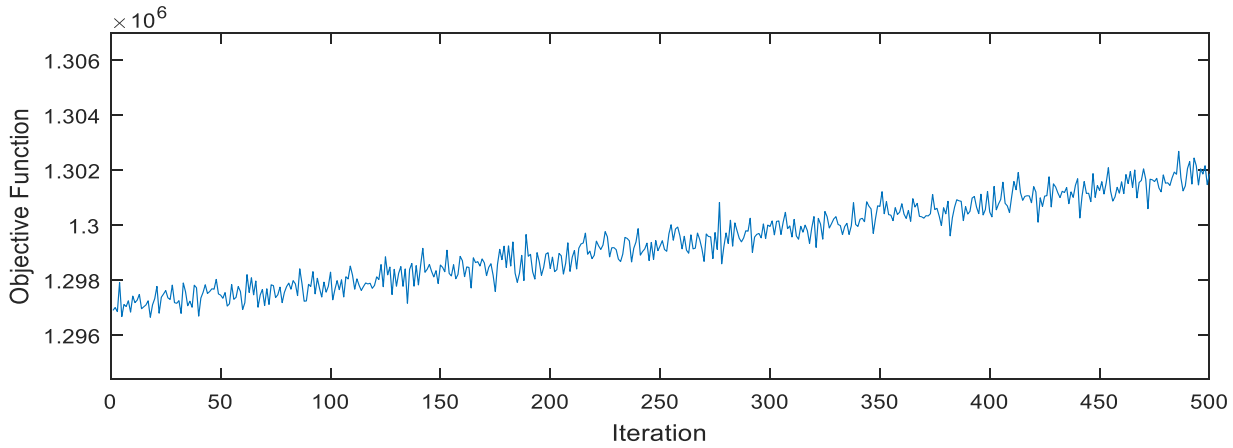


Figure 4.4: The objective function of the AM algorithm running on CPU after 20,000 GPU iterations

Transmission data issue: The clinical real data used in our previous are complicated, due to noise and incompletely known preprocessing. Therefore, we simulated noiseless transmission data from the NCAT phantom and reconstructed it with our implementation of the AM algorithm. Figure 4.5 shows objective function of the AM algorithm reconstructed from the simulated NCAT phantom. The objective function also increases after approximately 1000 iterations.

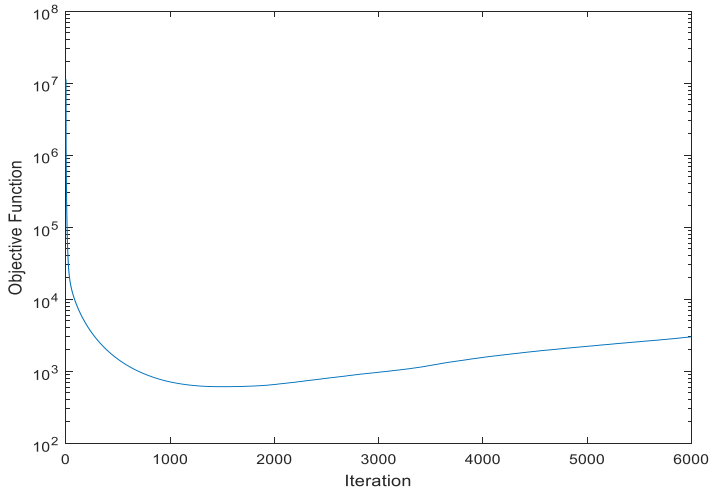
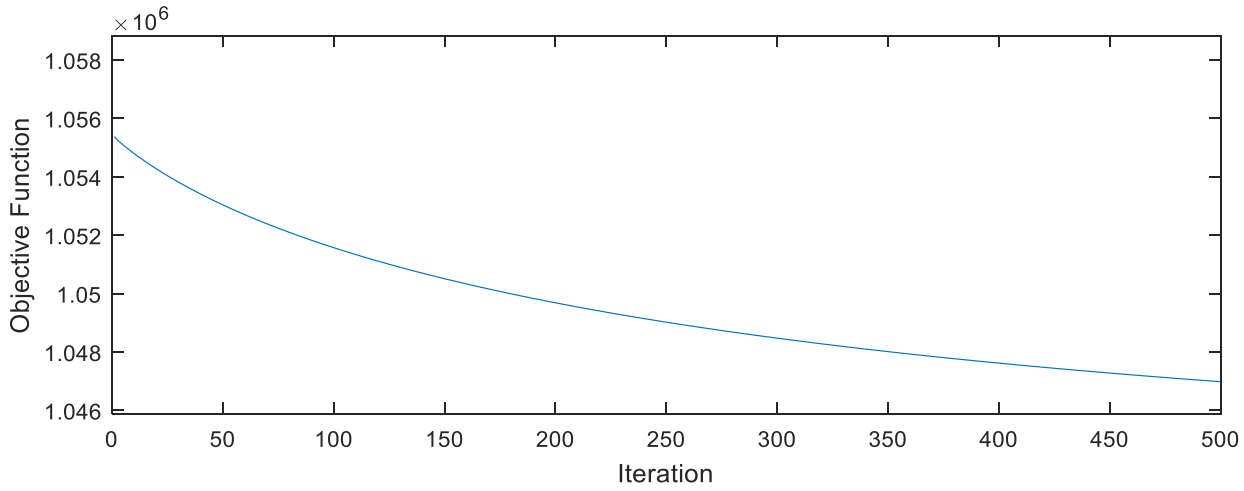


Figure 4.5: The objective function of the AM algorithm for NCAT-simulated data over 4700 iterations

Arithmetic Precision: Since the branchless distance-driven method contains numeric accumulation and back-accumulation processes, a lot of summation processes are required in our implementation, which is different from other implementations of the projections and backprojections. The summation is sensitive to both the order of execution of the processes and to their precision. Therefore, reaching the limit of precision should be obvious. A double-precision AM algorithm running on the a CPU was then implemented and the objective function between the different sets of code was analyzed. Considering the run time of the AM algorithm on a CPU, we used the output image of the single-precision AM algorithm on the GPU after 20,000 iterations as the initial condition, and ran the double-precision code on a CPU and single-precision code on GPUs. The results are shown in Figure 4.6. The plot of the result from the single-precision code on the CPU is also included as a reference. The objective function of the double-precision CPU implementation is decreasing. It differs from both the single-precision CPU and GPU implementations. We then concluded that the precision is the key issue. However, from Figure 4.6, we observed that the initial objective values are different, even if we used the same initial conditions. Further discussion of this issue is given in Section 4.1.2.



(a)

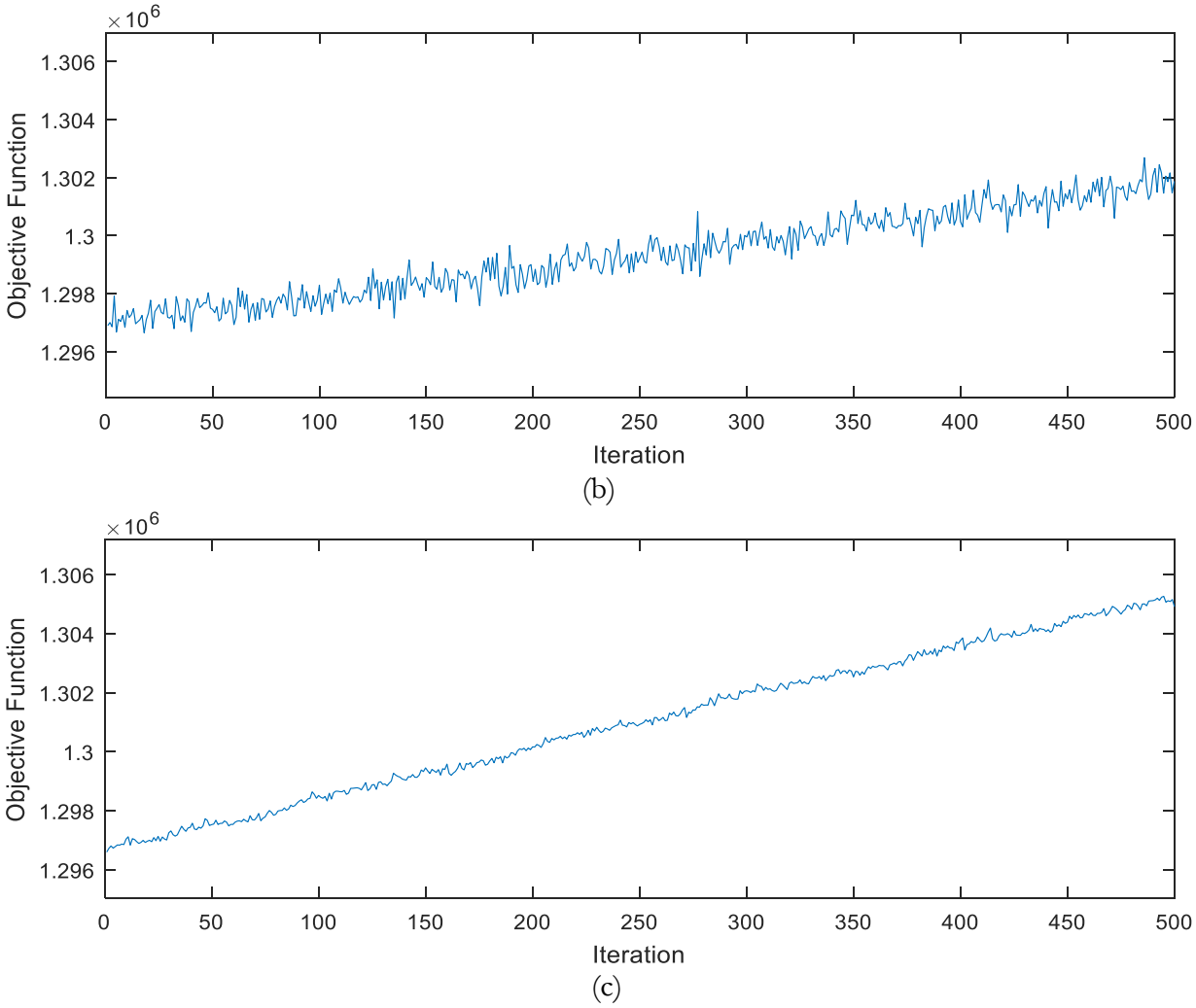


Figure 4.6 The objective function generated by (a) double-precision code on CPU (b) single-precision code on CPU, and (c) single-precision code on GPU

4.1.2 Code Inconsistency in Single and Double Precision

Analysis of the problem of the objective function increasing introduces a new problem: the different initial objective value between single-precision code and double-precision code. From figure 4.6, with the same initial condition, the initial value of double-precision code is $1.055*10^6$, while the initial value of both CPU and GPU single-precision code is 1.297×10^6.

4.2 Analysis

To solve the different-initial-value problem, all the procedures that compute I-divergence should be analyzed. The I-divergence is given by

$$I(d||g) \triangleq \sum_i \left[d_i \log\left(\frac{d_i}{g_i}\right) + g_i - d_i \right]$$

where d is the transmission data and g is predicted data based on the current image.

The computation of the initial value has two steps: projection and tje I-divergence computation. We computed the objective function of the same projected data in single and double precision. The similarity between the double-precision and single-precision I-divergence computational results shows that the I-divergence computation is not the problem. Therefore, we assumed that the projection of the initial image reached the limit of precision after thousands of iterations. In order to test our hypothesis, different initial images were used to compute different objective values in single and double precision. If our assumption is true, there should be a minor difference of I-divergences between single-precision and double-precision code when the objective function is decreasing. Table 4.1 shows the objective values of single-precision and double-precision implementations with different initial conditions

Table 4.1: The objective values of single-precision and double-precision code with different initial conditions

Initial objective value	Air	FDK	CG 100iter	AM 3000iter	AM 20000iter
Double precision	7,422,269,687	126,926,952	2,325,657	1,055,807	1,055,380
Single precision	7,422,269,687	126,927,251	2,325,683	1,058,700	1,297,048

As indicated in Table 4.1, the difference of the initial objective value between single and double precision is minor when the initial condition is zeros (air), FDK and the output image of 100 iterations of the conjugate descent (CG) algorithm. After 3,000 iterations of the AM algorithm, the difference increased, which indicates, during the projection process, some pixels have already reached the limit of precision.

With the AM algorithm iterating, the error is accumulating and an increasing numbers of projected pixels reached the limit of precision. The limit of precision in projection process leads to the inconsistency of objective function between single-precision and double-precision code. Moreover, the initial value of double-precision code is approximately the same as the minimum objective value that single-precision code could reach, which raises another question: in our single-precision implementation, while the objective function is increasing, is the reconstructed image approaching the optimal solution, or departing the optimal solution?

To solve this problem, we use the double-precision objective function as the distance between the reconstructed image and the optimal solution. The blue curve in Figure 4.7 shows the I-divergence of single-precision code for the NCAT simulated data, while the red curve is the double-precision I-divergence of the single-precision-reconstructed image. In other words, a set of reconstructed images is generated by a single-precision AM algorithm. Then, single-precision and double-precision codes are utilized to plot the single-precision and double-precision objective function of the set of reconstructed images, as the curves in Figure 4.7, respectively.

In Figure 4.7, the double-precision objective function is still decreasing while the single-precision objective function is increasing, which means, regarding the double-precision objective function as the distance measure, the single-precision-reconstructed image is not departing from the optimal solution as the single-precision objective function would seem to indicate.

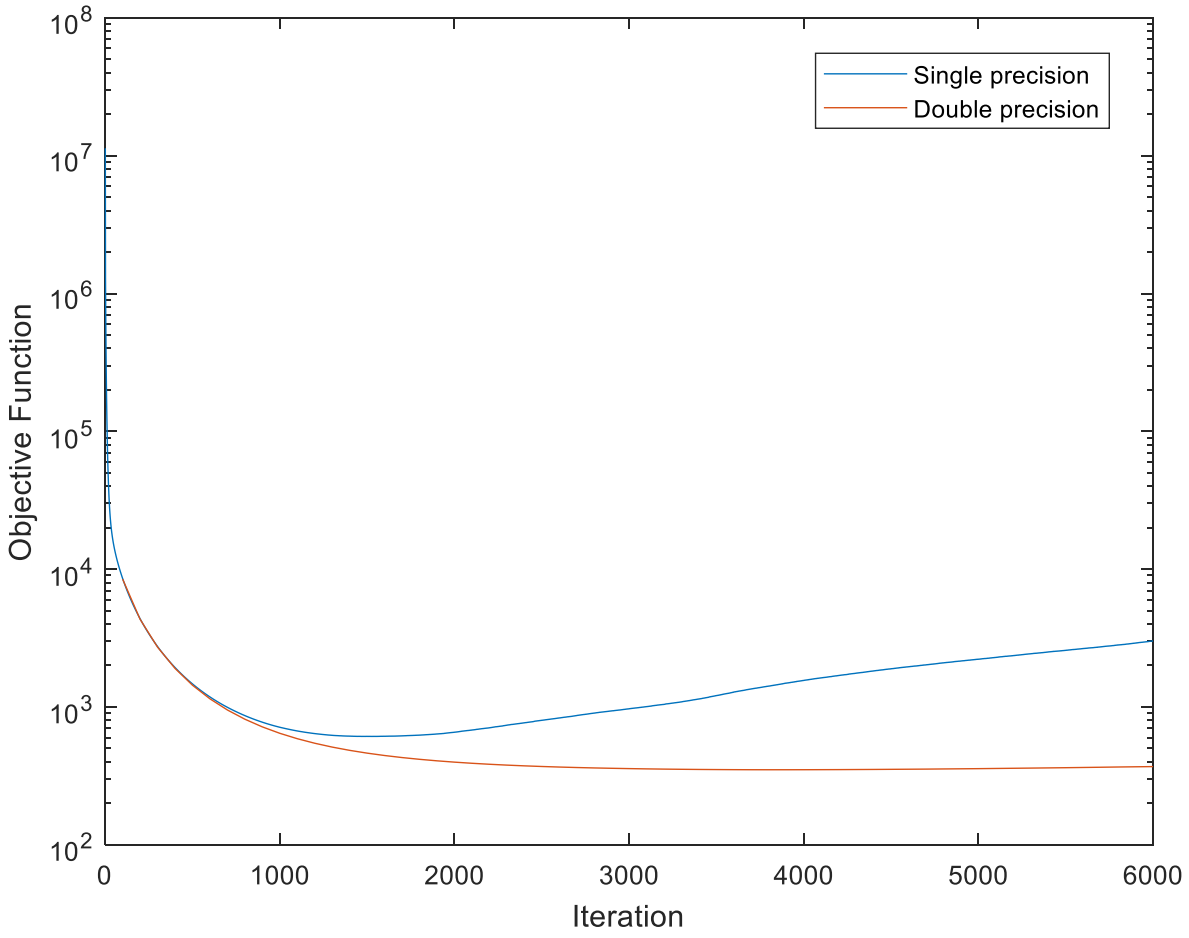


Figure 4.7: Objective function of single-precision AM algorithm for NCAT simulated data and its corresponding double-precision-computed objective function

Since the double-precision code can also be affected by the precision exhaustion issue, a more direct measure of distance is introduced. Differing from the clinical data, simulated NCAT data enable us to assess the difference between the test image and truth directly. Figure 4.8 shows the root mean square error (RMSE) between the reconstructed image and the NCAT phantom. Compared to the single-precision objective function, which starts increasing at approximately the 1000th iteration, the trend of RMSE plot is decreasing.

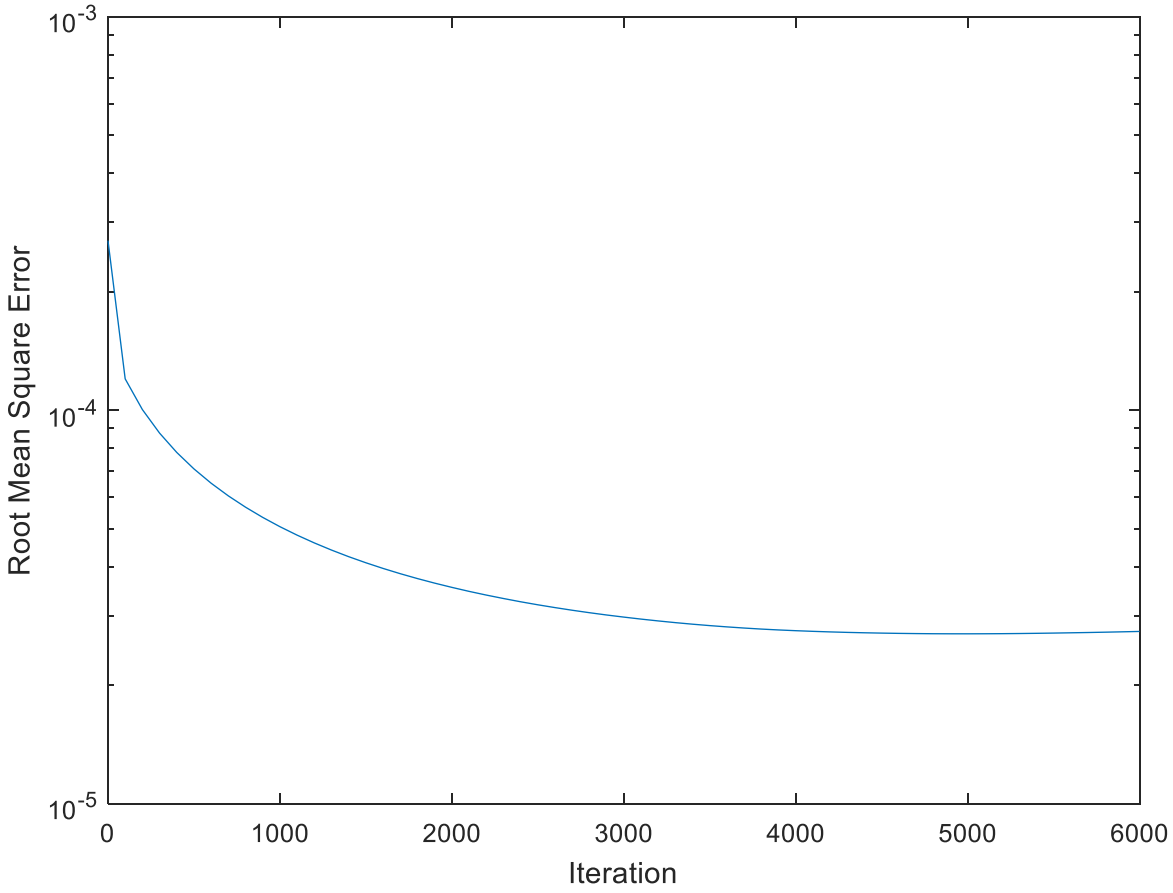


Figure 4.8: RMSE between the reconstructed images and truth

4.3 Conclusion

In Section 4.2, we conclude that the limit of precision in projection, as well as the numeric summation processes in the branchless DD method, lead to the inconsistency between single-precision and double-precision code. Due to this limit, the single-precision objective function may not correctly reflect the distance between the truth and the reconstructed images near the optimum. However, the single-precision objective function is still significant when we are far away from the optimal solution.

Moreover, limited by the precision, the objective function is supposed to eventually vibrate up and down about its asymptotic value. More experiments are still required to further explain the relationship between projection process and the non-stop growth of the objective function.

Chapter 5

Optimization of Parameters

In our graphical user interface, the algorithm is supposed to choose algorithm parameters automatically according to the clinical demand. In other words, if the "head" option is chosen, our application should set parameters automatically, such as λ , δ and the number of iterations. In this chapter, we will discuss methods for choosing parameters to satisfy the intentions of clinical researchers.

5.1 Experimental Program

The goal of CT reconstruction is translating the transmission data from the CT scanner to images that can be easily understood in medical practice. Based on a function quantifying the performance of our algorithms with specified parameters, we could find the optimal parameters that achieve the best image quality.

Figure 5.1 shows the designed process of the algorithm parameters optimization. The experiment is planned as follows:

1) Run the AM algorithm for a transmission data set with different sets of algorithm parameters λ and δ. The transmission data set should contain cases of data with or without lesions.

2) Apply image quality measure on the reconstructed image set generated in (1) to get a set of image performance with different algorithm parameters.

3) The combination of algorithm parameters with the best performance could be regarded as the optimal parameter combination among the parameters set for the specified environment corresponding to the transmission data set.

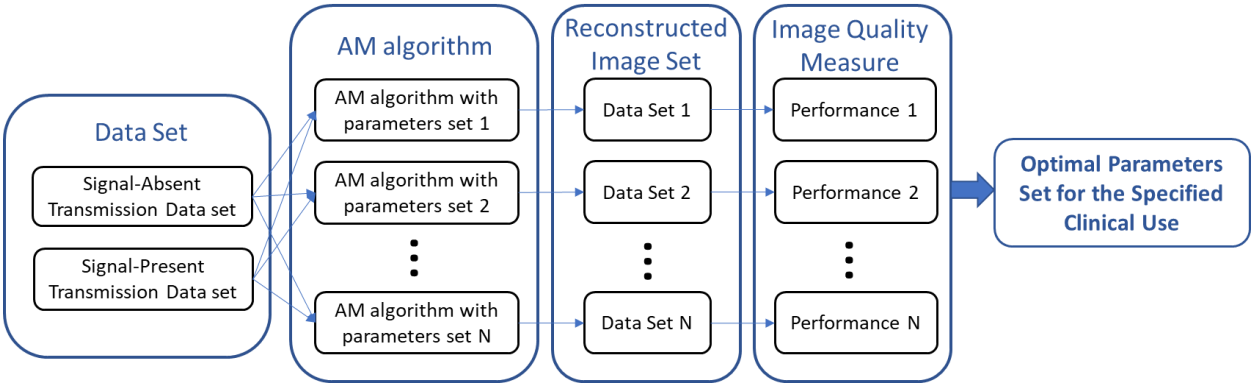


Figure 5.1: The designed process of the parameter-selecting experiment

5.2 Prediction of Human Observer Performance

In Section 5.1, we introduce a methodology for algorithm-parameters optimization. Based on our design, an image quality measure is important. One choice is the Hotelling observer, which is widely used to simulate the performance of human observer. Therefore, we can use the Hotelling observer to predict the performance of doctors on our reconstructed images.

5.2.1 Hotelling Trace Criterion (HTC)

The Hotelling Trace Criterion (HTC) is used to find a linear separator of two classes of objects, as well as a separability measure. Fiete RD, Barrett HH et al. [10] showed its relationship with the performance of human observer in 1987. The HTC is given by

$$\mathrm{HTC} = \mathrm{tr}\left(\boldsymbol{S_2^{-1}S_1}\right)$$

where "tr" denotes the trace of the matrix. S_1 is an inter-class scatter matrix given by

$$\boldsymbol{S_1} = \sum_{i=1}^{K} P_i\left(\bar{\boldsymbol{d}}_{\boldsymbol{i}} - \bar{\boldsymbol{d}}_{\boldsymbol{0}}\right)\left(\bar{\boldsymbol{d}}_{\boldsymbol{i}} - \bar{\boldsymbol{d}}_{\boldsymbol{0}}\right)^T$$

where $\bar{\boldsymbol{d}}_{\boldsymbol{i}}$ is the mean vector of the i^{th} class, $\bar{\boldsymbol{d}}_{\boldsymbol{0}}$ is the mean vector of all the objects from all classes and P_i is a priori probability of the i^{th} class.

$\boldsymbol{S_2}$ is an in-class scatter matrix given by

$$\boldsymbol{S_2} = \sum_{i=1}^{K} P_i \left(\sum_{j=1}^{N} p_j \left(\boldsymbol{d_{ij}} - \bar{\boldsymbol{d}}_{\boldsymbol{i}}\right)\left(\boldsymbol{d_{ij}} - \bar{\boldsymbol{d}}_{\boldsymbol{i}}\right)^T \right)$$

where $\boldsymbol{d_{ij}}$ is the j^{th} object in the i^{th} class, p_j is the a priori probability that $\boldsymbol{d_{ij}}$ appears in the i^{th} class, and N is the number of objects in one class. $\boldsymbol{S_1}$ assess the distances between classes, while $\boldsymbol{S_2}$ assess the mean distances between objects over all classes. The HTC is a scalar, positively quantifying the separability of a data set.

It was shown that merit of the receiver operating characteristic (ROC) curve of the HTC has a correlation coefficient of 0.988 with the merit of the ROC curve of humans on simulated liver CT images with or without tumors.

5.2.2 Channelized Hotelling Observer (CHO)

Based on the correlation between the merit of the HTC and human observers, the Hotelling observer (HO) is developed to predict the task performance of human observers in a signal-known-exactly (SKE) binary classification task, with channels to simulate the human visual system [11]. The weight and test statistic for the linear separator is given by

$$\boldsymbol{\omega} = \boldsymbol{S_2^{-1}}\left(\bar{\boldsymbol{d}}_{\boldsymbol{2}} - \bar{\boldsymbol{d}}_{\boldsymbol{1}}\right)$$
$$\lambda = \boldsymbol{\omega^T d_c},$$

where $\boldsymbol{d_c} = \boldsymbol{U^T d_t}$ is the channelized test image, $\boldsymbol{U}$ is the matrix representation of a set of the channel filters, $\bar{\boldsymbol{d}}_{\boldsymbol{2}}$ is the mean of the channelized signal-absent data and $\bar{\boldsymbol{d}}_{\boldsymbol{1}}$ is the mean of the channelized

signal-present data. Test statistic λ then could be used for classification with a threshold or computation of merit of the ROC curve.

In the CHO model, the test image is filtered through a set of channels with different frequencies. The channelized test image $\boldsymbol{d_c}$ is then generated as a vector of scalars from different channels. If we denote the test image as a vector with the size of N^2, and filter it with P channels, the dimension of the filtered test image is then reduced from N^2 to P, where $\mathrm{P} \ll N^2$.

One choice for the channel is the Gabor filter, which has been shown to simulate the 2-dimensional (2-D) response of simple cells in the visual cortex [12] [13]. The function of the Gabor filter could be expressed as

$$\mathrm{G}(x,y) = \exp\left(-\frac{4\ln 2((x-x_o)^2+(y-y_o)^2)}{{\omega_s}^2}\right) \cdot \cos\left(2\pi f_c\left((x-x_o)\cos\theta + (y-y_o)\sin\theta\right) + \beta\right)$$

where (x_o, y_o) is the spatial center, corresponding to the center of the lesion, ω_s is the width of the frequency band, f_c is the central frequency, θ is the orientation and β is the phase. The parameters of the Gabor filter, including ω_s, f_c, θ and β are selected with respect to properties of visual cells. In the implementation of Lifeng Yu et al. [14], they use six frequency band: [1/128, 1/64], [1/64, 1/32], [1/32, 1/16], [1/16, 1/8], [1/8, 1/4] and [1/4, 1/2]; five orientations: 0, 2π/5, 4π/5, 6π/5 and 8π/5; and two phases: 0 and π/2. The data dimension would be decreased to 60. The overall performance correlation on phantom-scanned data between the human observer and the CHO by Lifeng Yu et al. is 0.986.

5.2.3 Three-dimensional CHO

In Section 5.1.2, we introduced a channelized Hotelling observer as an estimate of the human observer for 2D images. In this section, to apply the CHO on our 3D AM algorithm, some implementations of three-dimensional CHO will be discussed.

In the current CT image diagnosis process, doctors may read the image either by slice, or in a volumetric 3D view. 3D CHO contains 2 types of models: volumetric CHO (vCHO) and multislices CHO (msCHO), depending on the way of the image reading is done. In these models, the test image is represented as $\boldsymbol{d_{t,3D}} = \left[\boldsymbol{d_{t,1}} \dots \dots \boldsymbol{d_{t,M}}\right]$, where M is the number of slices.

vCHO uses a bank of 3D filters to generate the channelized test data for Hotelling Observer [15]. In this case, $\boldsymbol{d_{c,v}} = \boldsymbol{U_v}^T \boldsymbol{d_{t,3D}}$, where $\boldsymbol{U_v}$ is the matrix of a bank of volumetric filters with different frequencies and $\boldsymbol{d_{c,v}}$ is the volumetric channelized test data filtered by these filters, whose dimension should equal the number of filters in $\boldsymbol{U_v}$. The weight for vCHO is given by

$$\boldsymbol{\omega_v} = \boldsymbol{S_{2,v}^{-1}}\left(\bar{\boldsymbol{d}}_{\boldsymbol{2,v}} - \bar{\boldsymbol{d}}_{\boldsymbol{1,v}}\right)$$

where $\bar{\boldsymbol{d}}_{\boldsymbol{1,v}}$ and $\bar{\boldsymbol{d}}_{\boldsymbol{2,v}}$ are the 3-D mean signal-present and signal-absent volumetric-channelized data, respectively, $\boldsymbol{S_{2,v}}$ is the in-class scatter matrix of volumetric-channelized data. Then the test statistic is given by

$$\lambda_v = \boldsymbol{\omega_v d_{c,v}}$$

Compared to 2-D channelized data that only contains contrast and structure in one slice, the volumetric channelized data would store the volumetric information, including contrast, structure and correlation between voxels. The vCHO process is shown in Figure 5.2(a).

Another way to present a 3-D CT images in clinical environment is presenting it as several slices of 2-D CT scans on film. msCHO simulates the process of multislice-based diagnosis. There are 2 models that could be implemented for msCHO. In the first model for msCHO, introduced by Mu Chen at el [16], two stages are required to compute the test statistic among images and slices, respectively. The filtered test image of the m^{th} slice is $\boldsymbol{d_{c,m}} = \boldsymbol{U_{2D}}^T \boldsymbol{d_{t,m}}$, where $\boldsymbol{U_{2D}}$ is the 2D channel matrix. The weight and test statistic among images is calculated with respect to a specific slice, specifically:

$$\boldsymbol{\omega_m} = \boldsymbol{S_{2,m}^{-1}}\left(\bar{\boldsymbol{d}}_{\boldsymbol{2,m}} - \bar{\boldsymbol{d}}_{\boldsymbol{1,m}}\right)$$
$$\lambda_m = \boldsymbol{\omega_m d_{c,m}}$$

Where the index $m \in [1, M]$ denotes a slice in test image, $\bar{\boldsymbol{d}}_{\boldsymbol{1,m}}$ and $\bar{\boldsymbol{d}}_{\boldsymbol{2,m}}$ are the mean signal-present and signal-absent 2D-channelized data of the m^{th} slice, respectively, $\boldsymbol{S_{2,m}}$ is the in-class scatter matrix of 2D-channelized data of the m^{th} slice. Other than the index standing for slice, this equation is the same as the equation in section 5.1.2. Let $\boldsymbol{\Lambda} = [\lambda_1, \lambda_2 \ldots \lambda_M]^T$ be the new variable vector. $\boldsymbol{\Lambda}$ is divided into two classes: signal-present slice $\boldsymbol{\Lambda_1}$ and signal-absent slice $\boldsymbol{\Lambda_2}$. The overall test statistic λ_{ms} is then given by

$$\lambda_{ms} = (\bar{\boldsymbol{\Lambda}}_{\boldsymbol{2}} - \bar{\boldsymbol{\Lambda}}_{\boldsymbol{1}})^{T} \boldsymbol{S_{2,\Lambda}^{-1}} (\bar{\boldsymbol{\Lambda}}_{\boldsymbol{2}} - \bar{\boldsymbol{\Lambda}}_{\boldsymbol{1}})$$

where $\bar{\boldsymbol{\Lambda}}_{\boldsymbol{2}}$ is the mean of λ of the signal-absent slices, $\bar{\boldsymbol{\Lambda}}_{\boldsymbol{1}}$ is the mean of λ of the signal-present slices, and $\boldsymbol{S_{2,\Lambda}^{-1}}$ is the intra-class scatter matrix of $\boldsymbol{\Lambda}$. The observer distinguishes the signal from the background among not only images, but also slices, using the HO test statistic. The model is shown in Figure 5.2(b).

In the second model, introduced by Ljiljana Platiša et al [16], the channelized test image is computed by a bank of multislice filters, $\boldsymbol{d_{c,ms} = U_{ms} d_{t,3D}}$, where $\boldsymbol{U_{ms}}$ is the channel matrix for integrated multislices. The weight, which is similar to the weight for vCHO, is given by

$$\boldsymbol{\omega_{ms}} = \boldsymbol{S_{2,ms}^{-1}}\left(\bar{\boldsymbol{d}}_{\boldsymbol{2,ms}} - \bar{\boldsymbol{d}}_{\boldsymbol{1,ms}}\right)$$

where $\bar{\boldsymbol{d}}_{\boldsymbol{1,ms}}$ and $\bar{\boldsymbol{d}}_{\boldsymbol{2,ms}}$ are the mean signal-present and signal-absent multislice-channelized data, respectively, $\boldsymbol{S_{2,m}}$ is the in-class scatter matrix of multislice-channelized data. The msCHO test statistic is then

$$\lambda_{ms} = \boldsymbol{\omega_m d_{c,ms}}$$

The process for this msCHO model is shown in Figure 5.2(c). Compared to vCHO, the msCHO models utilize the image information of pixel-direction and slice-direction separately. The second model is computationally simpler than the first one. Besides, according to Ljiljana Platiša et al., the second method result is closer to the ideal observer than the result of the first msCHO model when the data statistics are Gaussian.

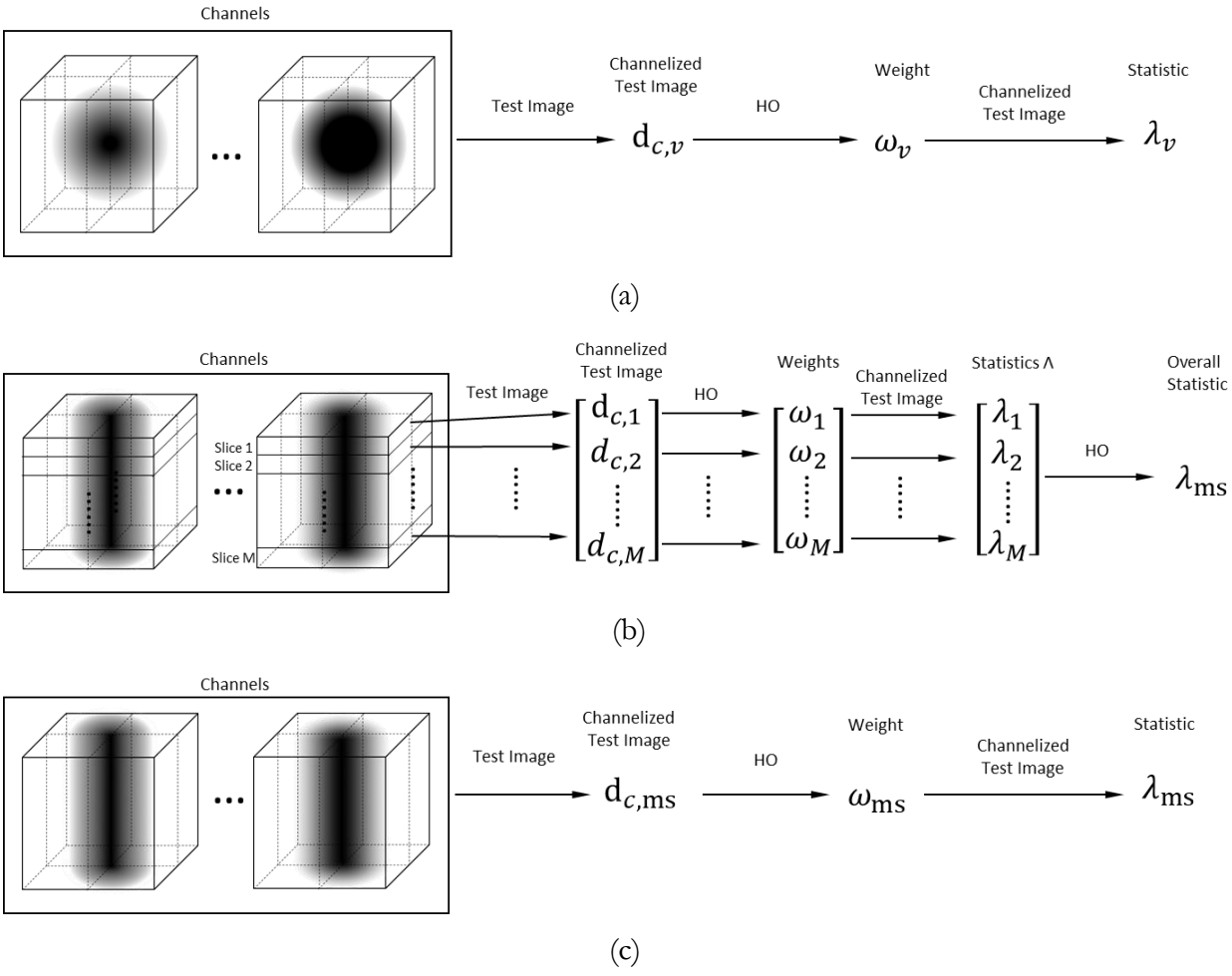


Figure 5.2: The process of (a) the vCHO (b) the first msCHO (c) the second msCHO

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In order to transition the GPU-accelerated CT reconstruction algorithm to a more clinical environment, this thesis discusses the acceleration methods, code-related problems, and methods for optimizing algorithm parameters in the future.

Using the FDK image as the initial condition significantly accelerates the convergence speed, not only by decreasing the initial objective value, but also by increasing the steps toward convergence. The no-OS-equivalent iteration is defined to assess the acceleration rate of the OS method. From the plots of no-OS-equivalent iterations, 145 OS performs the worst, 29 OS and 58 OS could be used for starting iterations, while 5 OS is stable and maintains the acceleration rate of 5X for 500 iterations. We also analyze the run time of our implementation with different numbers of OS. In 20 minutes, 29 OS is always the optimal selection for the AM algorithm, which achieves a speedup of 25X with respect to iteration. Then, code acceleration is implemented by modifying the CPU-version combination and accumulation process into a GPU-based version, which reduces the run time by 0.8 seconds per subset per iteration. Therefore, the run time of 29 OS is decreased from 71 seconds/iteration to 47.5 seconds/iteration. Moreover, we plot the gradient of the objective function versus the objective function with different numbers of OS, indicating that, to maximize the acceleration effect of the OS method, we should start with 29 OS, then switch to 5 OS at the 45^{th} iteration for the algorithm running more than 2200 seconds.

Moreover, we analyzed the problem of the increasing objective function value and the inconsistency of single-precision and double-precision code. After thousands of iterations, the AM algorithm reaches the limits of precision. Since the branchless distance-driven method requires numeric summation processes, the limit of precision greatly influences the projection processes with the error accumulating, which leads to an approximate 30% difference of the objective function value between

single-precision code and double-precision code. The limit of precision also limits the role of our objective function as a distance measure between the reconstructed image and the truth. We concluded that the increasing objective function does not indicate the failure of convergence of our implementation.

In Chapter 5, we built a prototype of the parameters optimization process and introduced a choice of the image quality measure. According to the high correlation between HO and the human observer, the 3D CHO with a bank of Gabor filters is introduced to assess the quality of reconstructed images.

6.2 Future Work

The runtime analysis provides us significant insight to achieve run time reduction to achieve 0.8 seconds per subset per iteration. However, from the runtime table, we observed that the backprojection procedure is 4 times slower than the forward-projection procedure. Since the projection and backprojection processes have the same number of steps, further investigation is required to determine this difference based on a more detailed timeline of processes in the GPUs.

In Chapter 4, we conclude that, limited by the precision of the projection processes, the objective value doues not represent the distance between the truth and the reconstructed image. However, the reason for the non-stop growth of I-divergence computed using single precision should be further investigated.

In Chapter 5, we introduced a prototype of a parameter-optimization experimental program. Feasibility research for this prototype is required, with respect to model performance and data acquisition. A bank of 3-D filters is also needed to simulate the 3-D response of the visual cortex. Moreover, since there is a tradeoff between image quality and the running time in the iterative reconstruction algorithm, we should find a method to optimize the number of iterations.

References

[1] ImageJ Features. Retrieved from https://imagej.nih.gov/ij/features.html. April 2018.

[2] Samit Basu and Bruno De Man. Branchless distance driven projection and backprojection. *SPIE 6065, Computational Imaging IV,* 60650Y, February 2006.

[3] Joseph A. O'Sullivan, Fellow, IEEE, and Jasenka Benac. Alternating Minimization Algorithms for Transmission Tomography. *IEEE transactions on medical imaging,* Vol. 26, No. 3, Pages 283-297, March 2007.

[4] Ayan Mitra, David G. Politte, Bruce R. Whiting, Jeffrey F. Williamson and Joseph A. O'Sullivan. Multi-GPU Acceleration of Branchless Distance Driven Projection and Backprojection for Clinical Helical CT. *Journal of Imaging Science and Technology,* 2017; 61(1), January 2017.

[5] D. B. Keesing, Description and usage guide for HECTARE v. 1.0, *Helical CT Advanced Reconstruction Engine*, 2011.

[6] X. Tang, J. Hsieh, R. A. Nilsen, S. Dutta, D. Samsonov, and A. Hagiwara, "A three-dimensional-weighted cone beam filtered backprojection (CB-FBP) algorithm for image reconstruction volumetric CT helical scanning," *Phys. Med. Biol.*, vol. 51, pp. 855–874, 2006.

[7] Daniel Hillis and Guy L. Steele, Data Parallel Algorithms. *Communications of the ACM Number 12,* December 1986, pages 1170-1183, 1986.

[8] Sangtae Ahn, J.A Fessler, D. Blatt, and AO. Hero. Convergent incremental optimization transfer algorithms: application to tomography.

[9] K. Lange, Convergence of EM image reconstruction algorithms with Gibbs smoothing. *IEEE Trans. Med. Imaging*, 9, 439-446, 1990.

[10] R. D. Fiete, H. H. Barrett, W. E. Smith, and K. J. Myers. Hotelling trace criterion and its correlation with human observer performance. *J. Opt. Soc. Am. A*, Vol. 4, No.5, 945-953, May 1987.

[11] Harrison H.Barrett, Jie Yao, Jannick P. Rolland and Kyle J. Myers. Model observers for assessment of image quality. *Proc. Natl. Acad. Sci.* USA, Vol. 90, pp. 9758-9765, November 1993

[12] Judson P. Jones and Larry A. Palmer. An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology,* Vol. 58, No. 6, December 1987.

[13] John G. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *J. Opt. Soc. Am. A,* Vol. 2, No. 7, July 1985

[14] Lifeng Yu, Shuai Leng, LingYun Chen and James M. Kofler. Prediction of human observer performance in a 2-alternative forced choice low-contrast detection task using channelized Hotelling observer. *Medical Physics,* Vol. 40, No.4, 041908, April 2013.

[15] Ljiljana Platiša, Bart Goossens, Ewout Vansteenkiste, Subok Park, Brandon D. Gallas, Aldo Badano, and Wilfried Philips. Channelized Hotelling observers for the assessment of volumetric imaging data sets. *J. Opt. Soc. Am. A,* Vol. 28, No. 6, Pages 1145-1163, June 2011.

[16] Mu Chen, James E. Bowsher, Alan H. Baydush, Karen L. Gilland, David M. DeLong, and Ronald J. Jaszczak. Using the Hotelling Observer on Multislice and Multiview Simulated SPECT Myocardial Images. *IEEE Transactions on nuclear science,* Vol. 49, No. 3, June 2002.